UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de

Ingeniería y Sistemas de Telecomunicación

PROYECTO FIN DE GRADO

# PLATAFORMA TERAPEÚTICA BASADA EN KINECT PARA NIÑOS CON PARÁLISIS CEREBRAL INFANTIL

*BELÉN DURO GONZÁLEZ*

Grado en Ingeniería Telemática

Julio de 2015


## ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

# PROYECTO FIN DE GRADO

**TÍTULO:** Plataforma terapéutica basada en Kinect para niños con parálisis cerebral infantil

**AUTOR:** Belén Duro González

**TITULACIÓN:** Grado en Ingeniería Telemática

**TUTORA:** Mª Luisa Martín Ruiz

**DEPARTAMENTO:** Departamento de Ingeniería Telemática y Electrónica

VºBº

**Miembros del Tribunal Calificador:**

**PRESIDENTE:** Irina Argüelles Álvarez

**TUTORA:** Mª Luisa Martín Ruiz

**SECRETARIO:** Iván Pau de la Cruz

**Fecha de lectura:**

**Calificación:**

**El Secretario,**

*"Todo esfuerzo tiene su recompensa"*

*Aquí está la mía.*

*AGRADECIMIENTOS*

*A mi familia, por su apoyo incondicional.*

*A mi tutora Marisa, por darme la oportunidad de ser parte de este proyecto tan bonito.*

# Resumen

El sistema SONRIE (Sistema de terapia, basadO en KiNect, paRa nIños con parálisis cErebral), realizado como Proyecto Fin de Grado por Dña. Estefanía Sampedro Sánchez, se desarrolló con el fin de permitir el proceso de rehabilitación de los músculos faciales en niños con Parálisis Cerebral Infantil (PCI). SONRIE se compone de una plataforma de juegos cuyo objetivo es lograr una mejora terapéutica en la musculatura orofacial de niños diagnosticados de PCI con edades comprendidas entre los 4 y los 12 años. El escenario de aplicación del sistema SONRIE son las escuelas de integración que tienen escolarizados alumnos diagnosticados con este trastorno.

La posibilidad de rehabilitación de los músculos faciales mediante tratamientos que se apoyan en el uso de sistemas telemáticos, junto con el empleo de tecnologías actuales (Realidad Virtual, Realidad Aumentada y Serious Games) supone una gran innovación en el entorno de la neuro-rehabilitación, entendida como el proceso de terapia que permite optimizar la participación de una persona en la sociedad, alcanzando un grado de bienestar óptimo.

El trabajo realizado en este Proyecto Fin de Grado pretende escalar el sistema SONRIE, mediante el análisis, diseño y desarrollo de un *Framework* encargado de facilitar, ampliar y validar el uso adecuado del sistema SONRIE en entornos escolares a través de la integración de nuevas tecnologías. La plataforma desarrollada en este proyecto, permite dotar de dinamismo y persistencia a la plataforma de juegos, ofreciendo a los usuarios de SONRIE (principalmente fisioterapeutas y rehabilitadores que trabajan en entornos escolares) un sistema de terapia para niños con PCI accesible vía web. En este Proyecto Fin de Grado se describe el conjunto de componentes software desarrollados con el fin de proporcionar un entorno web que escale el sistema SONRIE, convirtiéndolo en un sistema de terapia efectivo, completo y usable.

# Abstract

The SONRIE system (Sistema de terapia, basadO en KiNect, paRa nIños con parálisis cErebral), performed as a final project by Miss Estefanía Sampedro, was developed in order to allow the rehabilitation process of the facial muscles of children with Cerebral Palsy (CP). SONRIE consists of a gaming platform which aims to achieve a therapeutic improvement in the orofacial musculature on children diagnosed with CP aged between 4 and 12 years. The application scenario of the SONRIE system are the integration schools that have students diagnosed with this disorder.

The possibility of rehabilitation of facial muscles through treatments based on the use of telematics systems, together with the use of new technologies (Virtual Reality, Augmented Reality and Serious Games) is a great innovation in the neuro-rehabilitation environment, understood as the therapy process that optimizes the participation of a person in the society, reaching an optimum level of welfare.

The work done in this final project aims to scale the SONRIE system, through the analysis, design and development of a framework in charge of facilitating, extending and validating the proper use of the SONRIE system in school environments, through the integration of new technologies. The platform developed in this project, can provide dynamism and persistence to the gaming platform, offering to the SONRIE users (mainly physiotherapists and rehabilitators who work in school settings) a therapy system for children with CP accessible via web. In this final project are described the software components developed in order to provide a web environment that scales the SONRIE system, making it an effective, complete and usable therapy system.

# Índice de contenidos

# Índice de Figuras

# Índice de Tablas

# Lista de acrónimos

**AJAX:** Asynchronous JavaScript And XML

**API:** Application Programming Interface

**AU:** Animation Units

**BD:** Base de Datos

**CP** Cerebral Palsy

**CSS:** Cascading Style Sheets

**DAO:** Data Access Object

**DTO:** Data Transfer Object

**EJB:** Enterprise JavaBeans

**FK:** Foreign Key

**HTML:** HyperText Markup Language

**HTTP:** HyperText Transfer Protocol

**Java EE:** Java Enterprise Edition

**JDBC:** Java Data Base Connectivity

**JSON:** JavaScript Object Notation

**PC:** Parálisis Cerebral

**PCI:** Parálisis Cerebral Infantil

**PK:** Primary Key

**POO:** Programación Orientada a Objetos

**REST:** Representational State Transfer

**RV:** Realidad Virtual

**SGBD:** Sistema Gestor de Base de Datos

**SOAP:** Simple Object Access Protocol

**SONRIE:** Sistema de terapia, basadO en KiNect, paRa nIños con parálisis cErebral

**SQL:** Structure Query Language

**SSOO:** Sistemas Operativos

**UML:** Unified Modeling Language

**URI:** Uniform Resource Identifier

**URL:** Uniform Resource Location

**WS:** Web Service

**WSDL:** Web Service Description Language

**XML:** eXtensible Markup Language

**SSL:** Secure Sockets Layer

# 1. Introducción

En la actualidad son numerosos los sistemas telemáticos orientados a la prestación de servicios médicos. Además, el desarrollo de nuevas tecnologías, en los últimos años, ha permitido un gran avance en los campos de la telemedicina y la e-salud, principalmente en la generación de nuevos servicios orientados a proporcionar acciones terapéuticas o rehabilitadoras en niños. Los desarrollos actuales permiten introducir contenidos dinámicos basados en juegos, proporcionando un entorno lúdico, motivante y adecuado para la consecución de objetivos terapéuticos diversos [LUN13]. El presente Proyecto Fin de Grado se centra en la utilización y mejora de una plataforma de juegos destinada a facilitar el proceso de rehabilitación orofacial en niños con parálisis cerebral infantil (PCI).

Existen numerosos sistemas que permiten la rehabilitación motora de niños con PCI, pero la mayoría de ellos se centran en movimientos corporales, dejando a un lado los movimientos faciales. Por ello surge el antecedente inmediato de este proyecto, el sistema SONRIE (Sistema de terapia, basadO en KiNect, paRa nIños con parálisis cErebral) desarrollado por Estefanía Sampedro [SAM15]. SONRIE se compone de juegos, diseñados en colaboración con especialistas en este grupo de trastornos, enfocados a la rehabilitación facial de niños con parálisis cerebral (PC). Cada juego se centra en la realización de una acción en la que intervienen varios conjuntos de músculos faciales, en concreto, subida de ambas cejas, soplido, beso y sonrisa. Con el sistema SONRIE se pretende explorar y trabajar los músculos encargados de los movimientos comentados para lograr una mejora en los procesos de deglución, gesticulación y del habla en niños con PCI (ver Anexo A).

El presente Proyecto Fin de Grado surge por la necesidad de mejorar el sistema SONRIE con el fin de ofrecer las herramientas necesarias para permitir una terapia efectiva y personalizada a cada niño que utilice el sistema. En concreto, se pretende dotar (a los fisioterapeutas y profesionales encargados de la mejora del estado muscular de los niños con PCI) de una plataforma software accesible desde cualquier dispositivo móvil, Tablet o PC que ofrezca un seguimiento de estos niños, así como la posibilidad de modificar ciertos parámetros relativos a los juegos con el fin de adecuarlos a las necesidades de cada niño en cada momento. Las primeras pruebas de validación de SONRIE tuvieron lugar en un centro de educación especial y demostraron que el sistema es capaz de detectar los movimientos faciales de estos niños y estimular la realización de movimientos, que en algunas ocasiones, son de difícil ejecución para estos niños. En este proyecto se recogerá el proceso seguido para añadir al sistema de juegos las funcionalidades necesarias para que los especialistas puedan modificar el comportamiento de los juegos adaptándose a cada individuo de manera personalizada. Permitiendo obtener una configuración totalmente individual de cada juego, según la terapia necesaria a seguir por el niño.

En el siguiente apartado se describen los objetivos del proyecto dentro del marco que se acaba de comentar.

## 1.1. Objetivos del proyecto

### 1.1.1. Objetivo general

El objetivo general de este proyecto es permitir a los especialistas el acceso a la configuración de los juegos y los usuarios del sistema SONRIE. Utilizando el modelo de tres capas se ofrecerá una plataforma web que se encargará de recoger los datos de los niños con PCI, ofreciendo una interfaz de consulta y configuración a los terapeutas, con el fin de proporcionar una terapia individual efectiva.

### 1.1.2. Objetivos específicos

Los objetivos específicos del presente proyecto son los siguientes:

- Estudiar el sistema SONRIE y analizar qué datos parametrizables son necesarios para asegurar la persistencia del sistema.
- Crear una base de datos que permita asegurar la persistencia de los datos del sistema SONRIE.
- Analizar y seleccionar la plataforma tecnológica más adecuada para el desarrollo del sistema.
- Analizar, diseñar y desarrollar una aplicación web que permita a los especialistas consultar los datos de cada paciente y modificar el comportamiento de los juegos para proporcionar una terapia adaptada a sus necesidades.
- Analizar, seleccionar y desarrollar el componente software más adecuado para la conexión entre los datos y su presentación, teniendo en cuenta que ha de estar basado en nuevas tecnologías y ofrecer al sistema escalabilidad en un futuro.

## 1.2. Estructura de esta memoria

En el siguiente apartado se va a describir la estructura de la presente memoria para un mejor acceso a su información.

La presente memoria pretende describir el proyecto realizado para obtener el sistema solución que cubre los objetivos anteriores. Para ello este documento se organiza en una serie de capítulos que realizan un recorrido descriptivo por el sistema implementado.

El primer capítulo una introducción al trabajo fin de grado realizado.

El segundo capítulo sitúa los antecedentes de este proyecto. También describe el sistema SONRIE.

El tercer capítulo expone el marco funcional y tecnológico en el que se desarrolla el proyecto.

El cuarto capítulo describe la solución propuesta en el proyecto dividiendo esta descripción en apartados dedicados a la selección de la plataforma tecnológica, las funcionalidades del sistema, la base de datos desarrollada, la arquitectura software de componentes de la aplicación, el diseño y construcción de las diferentes capas de software del framework, la gestión de errores que se realiza en el sistema y los archivos de configuración de la aplicación.

El quinto capítulo muestra los resultados obtenidos al probar y validar el sistema construido.

El último capítulo expone las conclusiones obtenidas tras el desarrollo, así como, una serie de futuras mejoras de la solución propuesta.

El resto del contenido de la memoria está constituido por anexos y demás material auxiliar como referencias bibliográficas, índices de acrónimos, etcétera.

# 2. Antecedentes

El desarrollo de nuevas tecnologías en los últimos años está proporcionando un entorno ideal para la generación de sistemas de apoyo que facilitan la rehabilitación de niños con alteraciones neurológicas. Encontramos numerosos estudios destinados a evaluar terapias basadas en realidad virtual (RV), entre ellos destaca el trabajo de Luna et al. en 2013 por utilizar la misma tecnología de captación que el presente PFG, Microsoft Kinect Xbox 360 [LUN13]. El objetivo del estudio era evaluar la plataforma software basada en videojuegos que ofrecía un sistema de apoyo interactivo a la rehabilitación de niños con PC. Se proporcionó este tratamiento a 11 niños con PC durante 8 semanas realizando pruebas también en niños totalmente sanos, concluyendo que el sistema era una herramienta perfectamente válida para la terapia de niños con parálisis cerebral. Además se demostró la efectividad de la terapia a largo plazo, ya que los resultados un año después eran prácticamente estables con respecto al primer estudio realizado. Estas conclusiones se basaron en la prueba Wilcoxon [BER93]. El estudio realizado por Sharanet et al. en 2012 fue el primero en analizar la terapia basada en la tecnología Wii Fit para la rehabilitación postoperatoria de niños con PC. Los resultados fueron positivos en la mejora del equilibrio y en otros aspectos como la motivación, la satisfacción y la cooperación durante la terapia [SHA12].

En 2011 se recoge una revisión de los sistemas de RV utilizados como herramientas de apoyo al tratamiento de niños con déficit de atención derivado de diferentes trastornos como hiperactividad, autismo y PC [WAN11]. Se obtuvieron resultados muy positivos a cerca de la utilización de los sistemas de RV en el hogar, destacando el aumento en el grado de participación del niño en las actividades propuestas y la reducción del tiempo empleado en desplazamientos a centros especializados. Además, recientemente se reconoció, tras la revisión y evaluación oportuna, el efecto positivo de la RV como herramienta útil y viable en la mejora de la movilidad de los miembros superiores de niños con PC [CHE14].

## 2.1. Sistema de apoyo terapéutico con Kinect para niños con parálisis cerebral infantil

Se han comentado algunos casos de estudio sobre sistemas basados en RV. Este apartado describe el antecedente inmediato de este proyecto, constituido por el Proyecto Fin de Grado titulado *"Sistema de apoyo terapéutico con Kinect para niños con parálisis cerebral infantil"* [SAM15]. En dicho trabajo se recoge el análisis, diseño e implementación del sistema SONRIE (Sistema de terapia, basadO en KiNect, paRa nIños con parálisis cErebral). SONRIE se compone de cuatro juegos pensados junto con expertos terapeutas para trabajar músculos faciales mediante la ejecución de gestos cotidianos: subida de ambas cejas, soplido, beso y sonrisa (ver Anexo A).

El sistema recoge los movimientos faciales del niño con PC durante el trascurso de los juegos empleando el dispositivo de captación Kinect 360 de Microsoft. Los movimientos realizados por el niño se analizan utilizando las facilidades que proporciona Kinect 360 para reconocer puntos faciales en cada iteración con el juego, obteniendo un resultado positivo o negativo. Para decidir este resultado el sistema utiliza procesos distintos. En los juegos de las cejas y la sonrisa, el resultado se obtiene calculando las distancias entre puntos faciales tanto en reposo como en el momento en el que se ejecuta el movimiento. Si la distancia actual es mayor que la distancia en reposo más un cierto umbral, se considera que el movimiento ha sido realizado correctamente. En cambio, en los juegos del soplido y del beso se utilizan herramientas proporcionadas en el SDK de Kinect. Estas herramientas denominadas Animation Units (AU) devuelven una respuesta entre -1 y 1 en función de cómo se han ejecutado ciertos gestos faciales [SAM15]. Por ejemplo, el sistema utiliza la AU2 donde el valor 1 corresponde a un movimiento con los labios estirados y el -1 con los labios redondeados y juntos, para reconocer el beso y el soplido.

La ejecución del sistema SONRIE a nivel usuario se presenta en los siguientes diagramas de actividad. El sistema comienza con la pantalla de inicio y una vez se ha detectado esqueleto del sujeto, se reproduce en bucle el primer juego, el de las cejas (Figura 1).

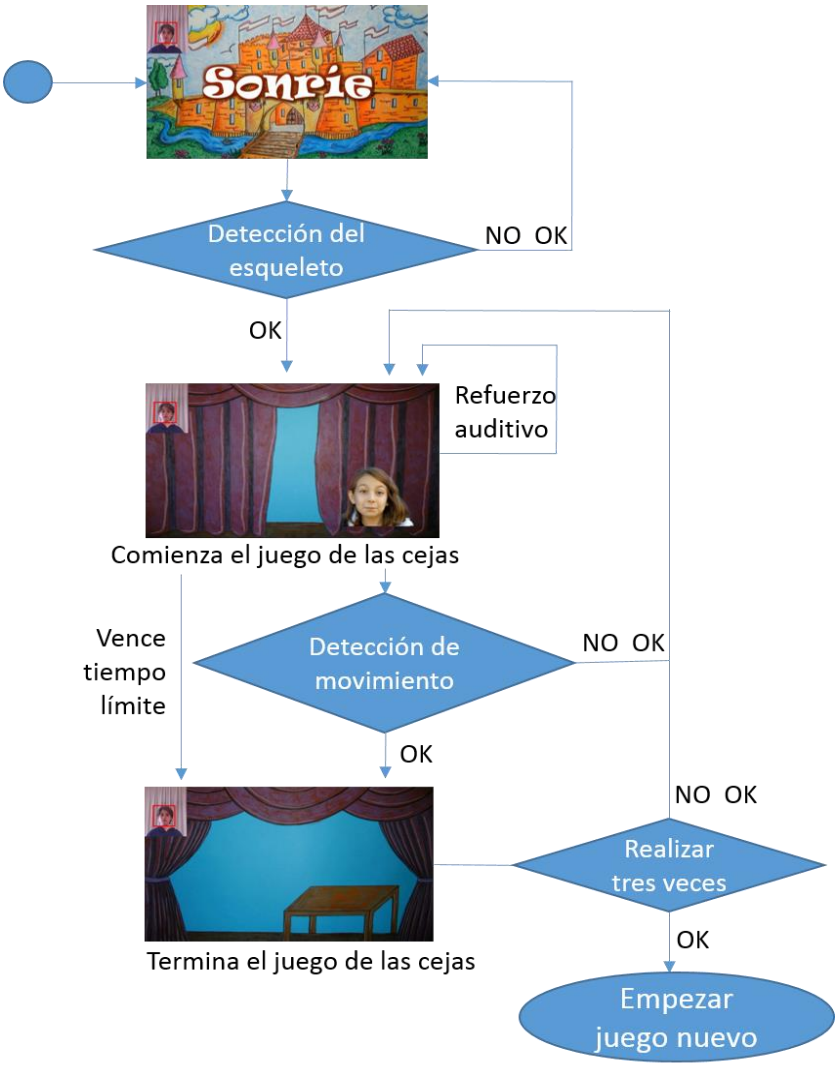


*Figura 1. Diagrama de actividad inicio y primer juego*

Cabe destacar el refuerzo positivo necesario para la aceptación del sistema por parte de los niños. Así, al comienzo de cada juego un vídeo explicativo les aporta el refuerzo sonoro necesario para asimilar de la acción a realizar, repitiendo aproximadamente a la mitad del tiempo establecido para cada intento. El sistema espera hasta que se detecte el movimiento o se exceda el tiempo límite establecido (15 segundos). Una vez se haya ejecutado el movimiento, sin importar el resultado, el sistema muestra una pantalla distinta con unos aplausos que permiten asegurar el refuerzo positivo necesario para captar su atención y proseguir con el juego.

Cuando se haya realizado tres veces el juego de las cejas, comenzará el juego del soplido cuya dinámica es exactamente la misma que en el juego anterior (Figura 2).

El sistema continúa ofreciendo refuerzos positivos a los niños. En este juego, una vez realizada la acción de soplar o vencido el tiempo, aparece en pantalla una vela apagada, con los aplausos de fondo. Si se realiza tres veces comienza el siguiente juego, el juego del beso.

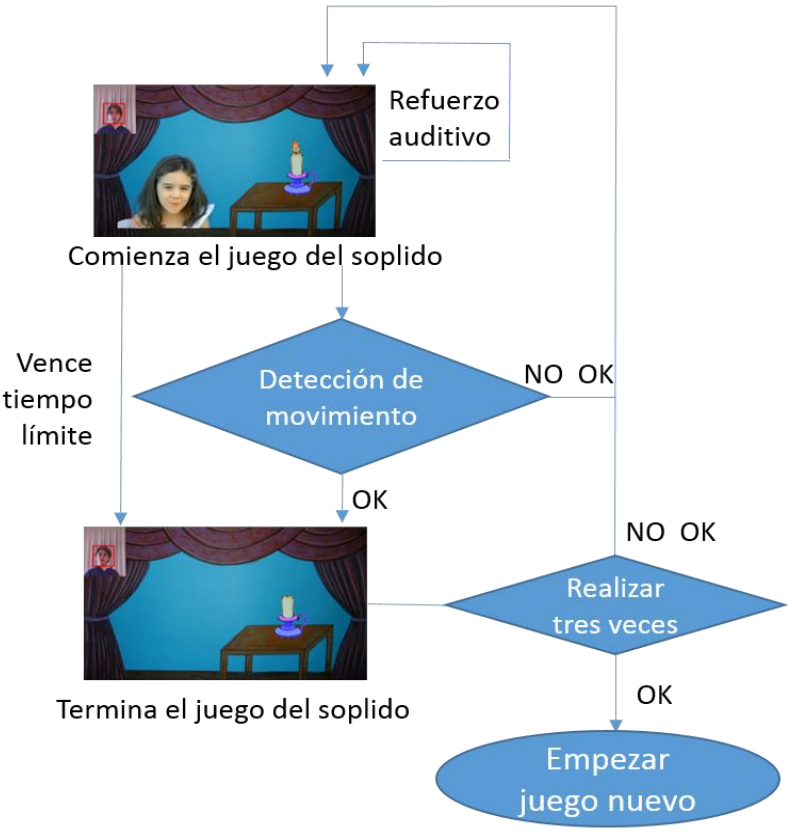


*Figura 2. Diagrama de actividad juego del soplido*

Sucede del mismo modo en el juego del beso (Figura 3). El juego debe reproducirse tres veces y cada uno de estos intentos tiene un tiempo límite de 15 segundos. Cuando se ha ejecutado el movimiento, suenan los aplausos y las guías del juego sonríen, dando comienzo al siguiente y último juego, el de la sonrisa (Figura 4)

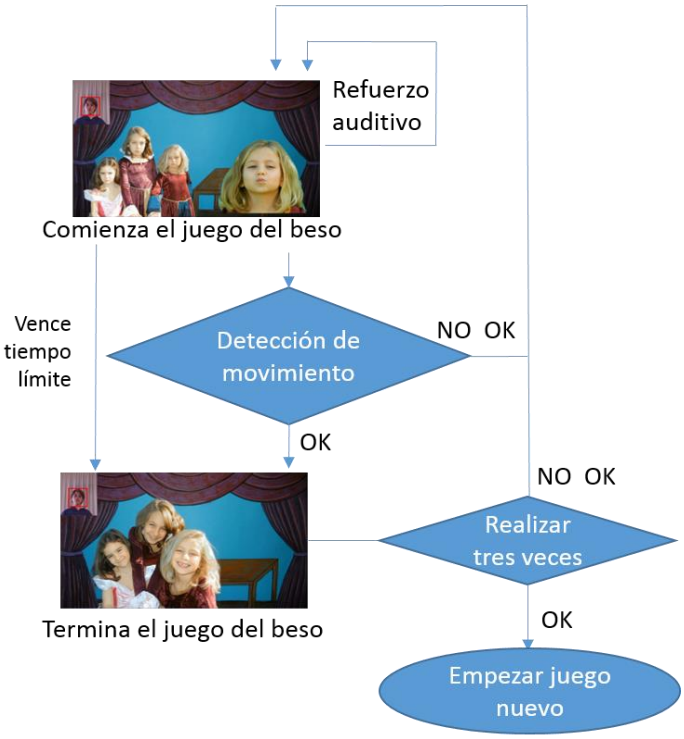


*Figura 3. Diagrama de actividad juego del beso*

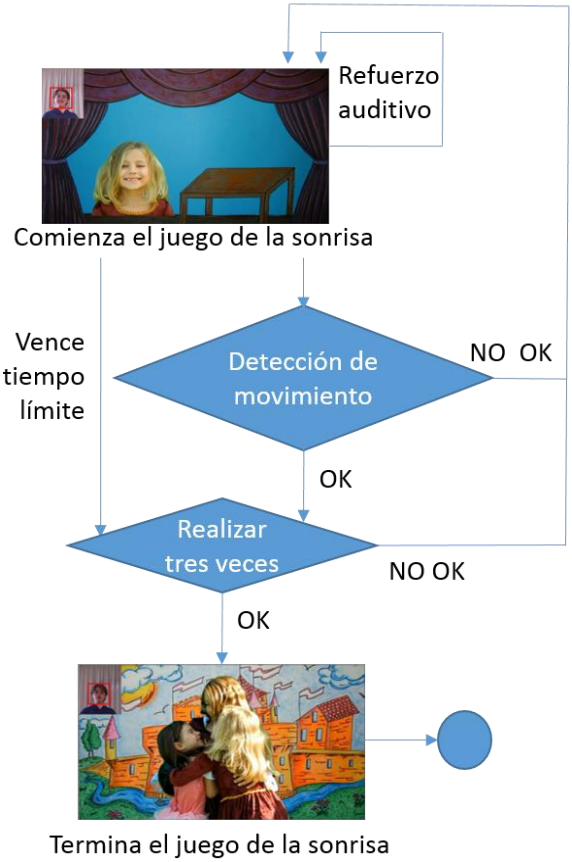


*Figura 4. Diagrama de actividad juego de la sonrisa y fin*

En este caso, por haber llegado a la ejecución del último juego, no se reproduce ningún vídeo si se excede el tiempo límite o se realiza el movimiento. En su lugar, permanece el vídeo de explicación del movimiento de la sonrisa y una vez se ha realizado tres veces se presenta la pantalla final en la que se reproduce un vídeo en el que se informa al niño de que el juego ha terminado y que lo ha realizado correctamente.

El sistema ha sido validado en un colegio de educación especial obteniendo resultados positivos en la aceptación del sistema por parte de los niños. Es por ello, que el presente PFG pretende facilitar a los terapeutas su labor a la hora de utilizar SONRIE, adaptando la dinámica de los juegos de cada niño a sus necesidades de cada momento y sin que este hecho conlleve la participación de un desarrollador o ingeniero en el proceso. Por el contrario, se pretende ofrecer una aplicación web intuitiva en la que los mismos especialistas interactúen con el sistema según sus criterios en la rehabilitación de los niños con PC.

A lo largo de los siguientes capítulos se mostrará el desarrollo realizado para conseguir los objetivos comentados para complementar el sistema SONRIE descrito en este capítulo. A continuación se detallan los contextos funcional y tecnológico en los que se ha desarrollado el sistema.

# 3. Contexto funcional y contexto tecnológico

Este apartado está dedicado a realizar una descripción de los contextos asociados al presente proyecto, lo que nos permitirá centrar la funcionalidad de la tecnología desarrollada, así como, describir la tecnología empleada para dar solución al sistema.

## 3.1.Contexto funcional

En un principio, el contexto vinculado al presente PFG es el entorno escolar, donde los terapeutas y especialistas facilitan los procesos de rehabilitación motora, únicamente corporal, de niños con PCI. Con SONRIE se incluye la rehabilitación motora de movimientos faciales, algo que todavía no se hacía, lo que implicaba problemas en estos niños a la hora de hablar o gesticular. Además, consideramos que el contexto de aplicación puede ser mucho más amplio, ya que este proceso de terapia puede completarse en el hogar y extrapolarse a centros terapéuticos a los que acuden estos niños para realizar rehabilitación.

## 3.2.Contexto tecnológico

El contexto tecnológico general, en el que se enmarca el presente proyecto, es el de una plataforma software distribuida con una interfaz de usuario web.

Una plataforma software distribuida es una evolución de la clásica arquitectura cliente-servidor en la que la parte servidora se divide en capas de software que pueden ejecutarse en diferentes servidores distribuyendo entre ellos el procesado de la información. Este hecho supone que, además de agilizar el tiempo de procesamiento, la distribución en componentes software permite escalar el sistema de manera sencilla en un futuro. Ya que, cada capa software se encarga de una funcionalidad completa, de manera que si fuera necesario ampliar el sistema, bastaría con ampliar la capa adecuada o, implementar e integrar una nueva capa software en lugar de modificar el sistema completo. Existen numerosos modelos de múltiples capas funcionales sin embargo, el más extendido y aceptado es el modelo de tres capas:

- Capa de presentación o interfaz de usuario.
- Capa de lógica de negocio o lógica funcional.
- Capa de datos.

En la actualidad existen dos plataformas software que siguen este paradigma:

- Plataforma Java edición Empresarial (Java EE)
- Plataforma Microsoft .NET

Estas plataformas tienen estructuras análogas, sin embargo, sus componentes se basan en lenguajes y tecnologías de desarrollo muy diferentes:

La plataforma Java EE es de libre distribución y código abierto. Utiliza tecnologías desarrolladas en el lenguaje de Programación Orientada a Objetos (POO) Java. Java EE es multiplataforma, lo que nos permite implantarlo en diversos Sistemas Operativos (SSOO) y servidores de aplicaciones. Esta plataforma ofrece varias especificaciones de Interfaces de Programación de Aplicaciones (API) muy útiles en el contexto que nos ocupa, como JDBC (Java Data Base Connectivity) o Servicios Web (WS), así como, los mecanismos necesarios para coordinar los diferentes componentes software de manera sencilla. Además, cabe destacar, que el lenguaje y la plataforma son de código abierto.

La plataforma .NET es un framework de Microsoft diseñado para SSOO Windows, aunque actualmente se está trabajando en la implementación de la plataforma para otros SSOO, como puede verse consultado la documentación relativa al Proyecto Mono [MON15]. La mayoría de las aplicaciones necesitan licencias de Visual Studio para desarrollar en .NET aunque hay herramientas gratuitas para el desarrollo web básico. Esta plataforma soporta un gran número de lenguajes de programación como C#, Visual Basic, C++, Perl o Python, ofreciendo herramientas para la integración y reutilización de componentes desarrollados en los diferentes lenguajes. También soporta la creación de servicios web basados en XML (eXtensible Markup Language) a través de SOAP (Simple Object Access Protocol) y WSDL (Web Service Description Language).

En la etapa de diseño se analizaron ambas plataformas con el fin de utilizar la plataforma tecnológica que más se ajustase al sistema a desarrollar. Esta elección parte de los lenguajes de implementación pero sobre todo de los componentes necesarios para ofrecer el servicio.

Si realizamos un análisis por capas, ambas plataformas admiten un cliente web que junto con el conjunto de páginas y módulos que componen la interfaz de usuario, tanto en el lado del cliente como del servidor, constituirán la capa de interfaz de usuario web. La comunicación entre esta capa y la capa de lógica residente en el servidor se realiza mediante el protocolo HTTP (HyperText Tranfer Protocol) en ambas plataformas. Este concepto supone asegurar un acceso universal a nuestro sistema, ya que una vez implementado y alojado en un servidor, en principio, cualquier usuario podrá acceder a él desde su navegador. Veremos que para dotar de seguridad al sistema será necesario proporcionar el acceso únicamente a usuarios del sistema.

La capa de lógica de la aplicación estará compuesta por diversos componentes software encargados de la lógica necesaria para ofrecer el servicio. Estos componentes, en ambas plataformas, deberán ejecutarse dentro de un contenedor perteneciente a un servidor web.

Como se ha comentado, ambos ofrecen APIs para la creación de WS. Este tipo de componentes surgieron para cubrir la necesidad de comunicar diferentes plataformas, y existen distintos tipos de WS basados en diferentes estilos de arquitecturas software. Será un objetivo principal de la etapa de diseño, decidir qué tipo de estilo de arquitectura se utilizará en la implementación de este componente. Los WS ofrecen un comportamiento muy adecuado a nuestro sistema como puede observarse en la Figura 5, en la que se muestra la interacción propia de un WS con un cliente.

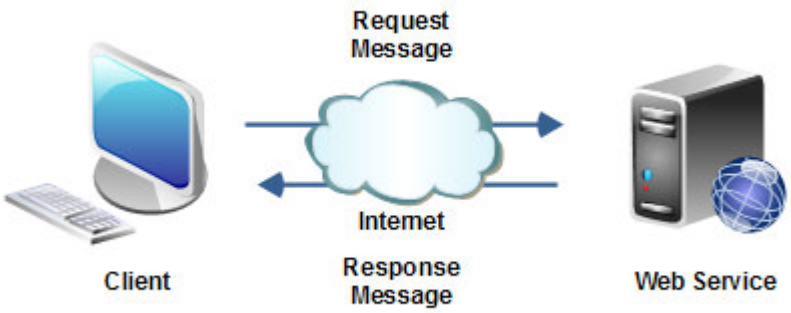


*Figura 5. Web Service [KEN14]*

La capa de datos, en general, queda constituida por una base de datos y un sistema gestor de base de datos independiente del servidor web de la capa de lógica, aunque podría ser algo tan sencillo como un sistema de ficheros. Lo fundamental es que la capa de datos ha de proporcionar persistencia al sistema.

# 4. Descripción de la solución propuesta

## 4.1. Plataforma tecnológica.

Partiendo del contexto tecnológico descrito se ha decidido utilizar la plataforma Java Enterprise Edition (Java EE). Esta decisión es fruto de un análisis de requisitos previos en el que están involucrados una serie de factores que la plataforma Java EE nos ofrece y que se detallan a continuación:

Java EE es un entorno independiente de la plataforma, centrado en la creación de desarrollos web. Además de la necesidad de emplear herramientas de libre distribución y nuevas tecnologías, se han tenido muy en cuenta los conocimientos adquiridos sobre esta plataforma, lo que permite centrarnos en el problema a resolver directamente. Una vez separadas las distintas partes del problema la plataforma Java EE, basada en POO, nos permite construir un objeto Java por cada módulo individual del problema a resolver. De hecho, la plataforma nos ofrece distintos módulos y patrones que se ajustan perfectamente al modelo de sistema concebido en la etapa de diseño. Además de modular, la arquitectura es distribuida y permite la reutilización de los módulos que componen el desarrollo. Estos hechos además de permitirnos albergar nuestros componentes en máquinas distintas implementando un sistema distribuido, permiten escalar el sistema fácilmente añadiendo nuevos módulos o reutilizando los componentes desarrollados con lo que los requisitos principales quedan cubiertos. La Figura 6 muestra los contenedores existentes en un servidor de aplicaciones Java EE mostrados en una arquitectura en capas.

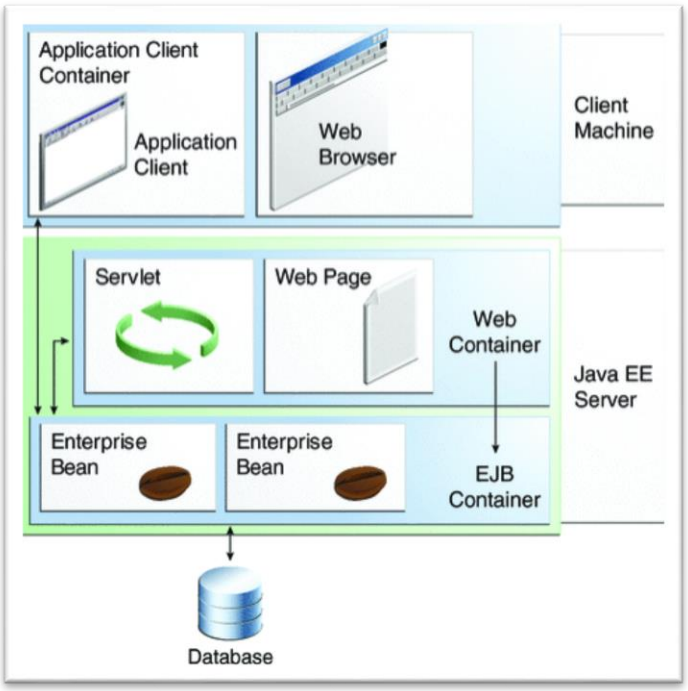


*Figura 6. Servidor Java EE y contenedores [ORA14]*

El sistema desarrollado se basa en este modelo general, sin embargo, se han introducido algunos cambios sobre esta arquitectura ya que algunos contenedores no son necesarios. Estas modificaciones junto con la estructuración en capas del sistema, pueden visualizarse en la Figura 7 que está basada en la Figura 6. Las modificaciones que se han introducido con respecto a la anterior son las siguientes:

En primer lugar, como puede verse en la Figura 6, existe un servidor Java EE compuesto por un contendor web y un contenedor de Enterprise JavaBeans (EJB). El servidor Java EE es un servidor de aplicaciones, lo que significa que es capaz de gestionar toda la lógica de negocio y acceso a datos incluyendo la comunicación con el cliente en diversos lenguajes, asegurando la disponibilidad de las aplicaciones desplegadas. Sin embargo, el contenedor de EJB, no es necesario en nuestro sistema, con lo cual se ha decidido sustituir estos componentes por otros denominados JavaBeans. Estos componentes permiten acceder directamente a ellos sin necesidad de un componente auxiliar de tipo proxy y no necesitan un contenedor para ejecutarse como los EJB. Puesto que con JavaBeans podemos implementar las funciones necesarias para la capa de acceso a datos, simplificando el tratamiento de los datos, en nuestro sistema no será necesario un contenedor de EJB para cumplir con los requisitos. Por tanto, únicamente necesitaremos un contendor web dentro de algún tipo de servidor. Existen otro tipo de servidores denominados servidores web que únicamente utilizan HTTP o SSL (Secure Sockets Layer) como protocolos de comunicación. En nuestro caso, dado que hemos eliminado la capa EJB, no tiene sentido emplear el servidor de aplicaciones Java EE. En su lugar, la solución más óptima consiste en emplear un servidor web. Se ha decidido emplear el servidor web libre más ampliamente utilizado, el servidor web Apache Tomcat. Sin embargo, si fue necesario utilizar el contenedor de Servlets Java para desplegar correctamente el sistema en el servidor.

Si bien es cierto que los componentes se han implementado en la plataforma Java EE descrita, una vez desarrollados se alojarán en un contenedor Tomcat dentro del servidor Apache Tomcat. Apache Tomcat es un servidor web multiplataforma desarrollado en Java e implementa varias especificaciones de Java EE. Utilizaremos como contenedor web de los componentes desarrollados, la solución que proporciona la Apache Software Foundation, el contenedor web Apache Tomcat [APA15]. Este contenedor se encarga de gestionar la ejecución y el ciclo de vida de los componentes web desplegados en él. Lo más interesante sobre Apache es que es de código abierto y software libre y, además, es ampliamente utilizado, su configuración es sencilla y existe mucha documentación disponible sobre este servidor web.

La siguiente diferencia con respecto a la Figura 6 es la inclusión de un componente muy actual, un Web Service (WS). Será necesario el uso de un Servlet con la información necesaria para que el contenedor Tomcat, donde alojaremos el sistema, pueda desplegar el WS junto con otro tipo de componentes encargados del acceso a los datos. Los servicios web son independientes de lenguajes de desarrollo, plataformas y protocolos de

comunicación, por tanto, el uso de un WS aporta interoperabilidad a nuestro sistema. Las tecnologías asociadas a este tipo de componentes se apoyan en estándares y protocolos basados en texto, por ejemplo, XML como formato de datos universal y HTTP como protocolo de transporte. Las arquitecturas orientadas a servicios implementadas a través de WS facilitan la comunicación y el acceso al contenido, así como, la interoperabilidad, usabilidad, reutilización y despliegue de componentes. El auge en el desarrollo de arquitecturas basadas en estos componentes y todas las ventajas mencionadas, han condicionado la elección de un WS Java como componente fundamental de la lógica de negocio del sistema.

En cuanto a la capa de datos, nuestro sistema utilizará el sistema gestor de bases de datos relacionales de software libre más extendido, el servidor de Base de Datos (BD) MySQL [ORA15]. MySQL es la BD de código abierto de mayor aceptación mundial. Además de ofrecer una gran facilidad de uso aporta un alto rendimiento y fiabilidad a esta capa del sistema.

En la siguiente figura pueden observarse las modificaciones realizadas sobre la anterior, así como la estructura en capas del sistema.

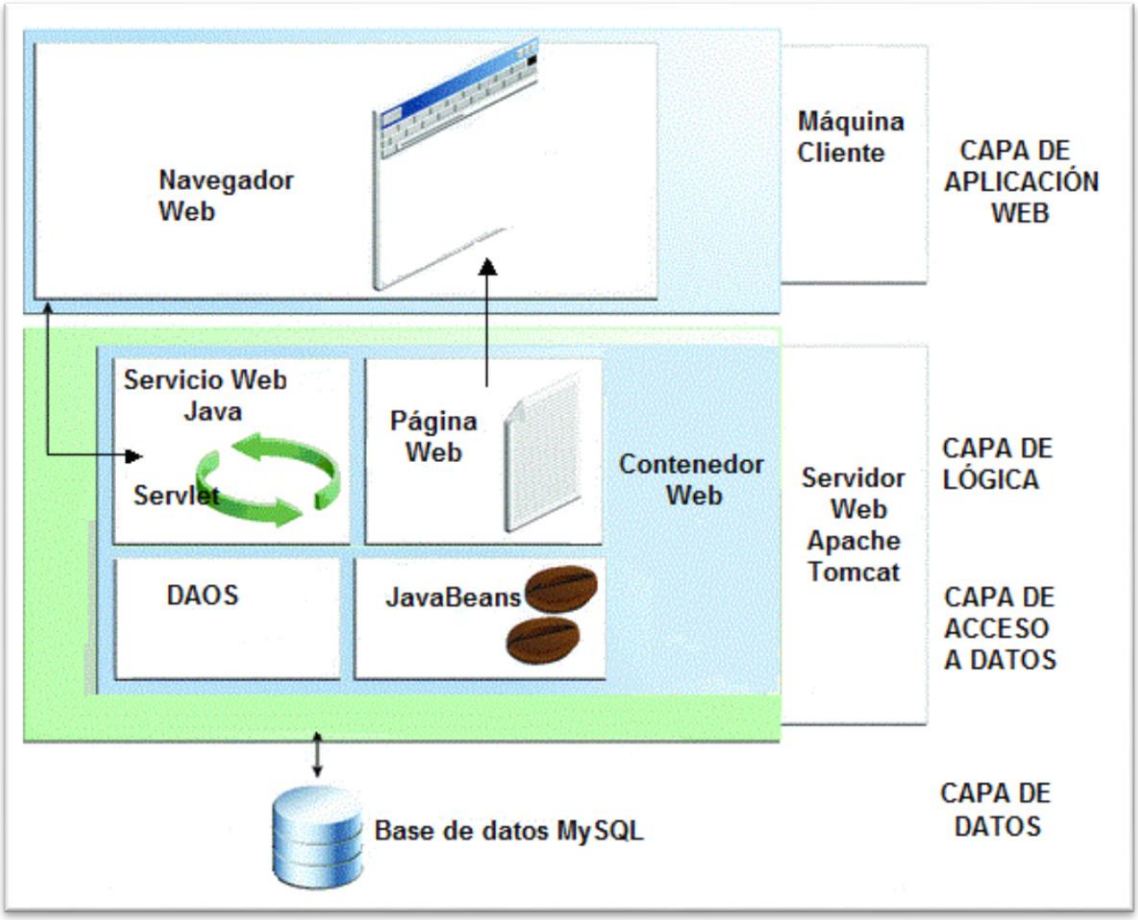


*Figura 7. Vista en capas del sistema SONRIE*

Utilizando la plataforma tecnológica descrita, se han diseñado e implementado un conjunto de componentes software según el modelo de tres capas, desacoplando el acceso a los datos de la lógica de negocio, por tanto, nuestro sistema cuenta con 4 capas distintas que explicarán con mayor detalle en los siguientes epígrafes.

En el nivel más bajo se encuentra la capa de datos compuesta por la base de datos MySQL encargada de almacenar toda la información persistente del sistema. Por encima, encontramos el nivel de acceso a datos encargado de obtener conexión con la base de datos y ejecutar sentencias sobre ella. El nivel de acceso se compone de los objetos DAO (Data Access Object) y los JavaBeans. El nivel superior, la capa de lógica de negocio, utiliza este nivel de acceso para ofrecer los datos necesarios a la aplicación web y actualizar la base de datos del sistema en función de los cambios introducidos vía web. Como vemos, el flujo de peticiones circula siempre de arriba hacia abajo, es decir, del cliente hacia el nivel de datos y nunca al revés. Esto significa que el nivel de acceso a datos y el nivel de datos nunca envían peticiones hacia el resto de niveles.

En el apartado siguiente se muestra la arquitectura de componentes simplificada del sistema introduciendo los componentes más importantes de cada una de las cuatro capas descritas, así como sus funcionalidades.

### 4.1.1. Arquitectura simplificada del sistema

En la Figura 8 puede visualizarse la arquitectura simplificada del sistema. Por un lado en una máquina cliente se encuentra el nivel de aplicación constituido por una página web HTML5 (HyperText Markup Language) y un conjunto de tecnologías web que se describirán más adelante. Mediante el protocolo HTTP este nivel se comunica con la capa de lógica y esta última con la de acceso a los datos. Aunque estas capas son distintas entre sí, los componentes que las constituyen residen en el mismo servidor. La capa de acceso se comunica con el nivel de datos utilizando la API JDBC (Java Data Base Connectivity). La BD que constituye el nivel de datos, puede residir en la misma máquina o en otra distinta, pues este hecho no afecta al funcionamiento global del sistema ya que como vemos se ha diseñado un sistema totalmente distribuido y flexible.

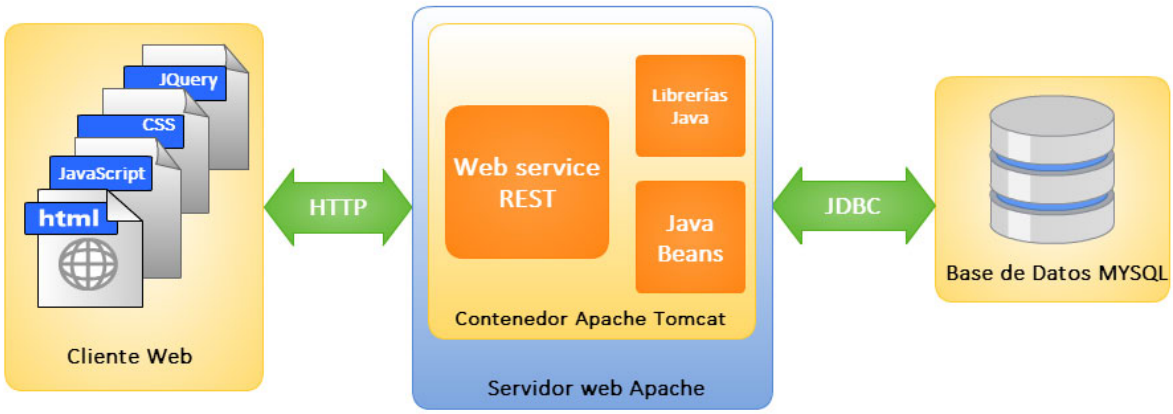


*Figura 8. Arquitectura simplificada del sistema*

A continuación analizaremos cada una de las capas, en primer lugar de una manera más general, obteniendo mayor detalle en los siguientes capítulos.

## Capa de acceso a datos

Esta capa se encuentra entre la capa de datos y la capa de lógica. Normalmente se implementaría en la capa lógica pero se ha decidido desacoplar el acceso a los datos de la lógica de negocio asegurando el sistema sea perfectamente escalable y flexible. Se han centralizado las funciones de acceso en esta capa, otorgando de mayor transparencia al sistema, debido a que la capa de negocio no conoce el destino de los datos ni ningún detalle acerca de la base de datos.

## Capa de lógica

Esta capa está constituida por un servicio web que utiliza los objetos de acceso a datos para obtener los datos de la base de datos, encapsulados en objetos Java. En primer lugar, este nivel se encarga de obtener la conexión con la base de datos utilizando el objeto de acceso a datos adecuado. Una vez ejecutada la sentencia, se ocupa de generar la respuesta en un formato válido para la aplicación. La comunicación entre el nivel de aplicación y la capa de lógica se realiza en base a URIs (Uniform Resource Identifier). Definiremos como se construyen estas URIs en el capítulo dedicado a la lógica de negocio de nuestro sistema.

## Capa de aplicación web

Esta capa constituye la interfaz con nuestros clientes, está diseñada en HTML5 para asegurar la accesibilidad de la aplicación desde cualquier dispositivo. La aplicación web se encarga de consumir el servicio web y de mostrar, de forma asíncrona, la información demandada en peticiones HTTP.

## Capa de datos

La capa de datos está constituida por un conjunto de tablas pertenecientes a una base de datos de tipo MySQL y destinadas a albergar toda la información necesaria para asegurar

el correcto funcionamiento del sistema. La selección de qué información es necesaria para asegurar la persistencia del sistema se detalla en el apartado dedicado a la base de datos del sistema.

Así, una vez desarrollado cada componente según está arquitectura, se procederá a desplegar el sistema SONRIE según la Figura 9. El sistema residirá en un servidor de la Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST) perteneciente a la Universidad Politécnica de Madrid (UPM).

Los distintos *profesionales* podrán acceder a través de Internet a la página web del sistema para disponer de los servicios de consulta, modificación, alta y baja detallados en el siguiente apartado. Además, existirá un *usuario administrador* encargado de otras funciones, como por ejemplo, el alta de nuevos juegos o profesionales en el sistema. En el caso de la inclusión de nuevos juegos y otros aspectos que signifiquen la modificación de la plataforma de juegos implementada, será necesario un *desarrollador web* que amplíe el sistema.

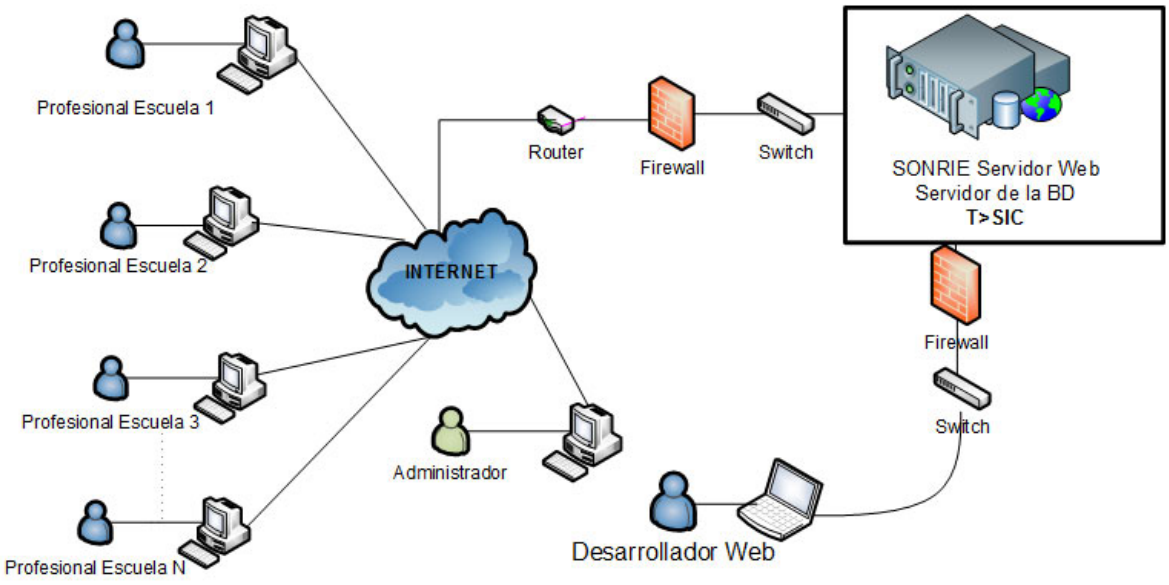


*Figura 9. Diagrama de despliegue del sistema*

## 4.2. Funcionalidades del sistema

Una vez descrita la arquitectura y plataforma tecnológica utilizadas para desarrollar el sistema, se analizarán las funcionalidades que implementa. Para describir estas funcionalidades se recurrirá al lenguaje de modelado de sistemas software más conocido y utilizado, el Lenguaje Unificado de Modelado (Unified Modeling Language, UML).

En concreto, se utilizarán diagramas de casos de uso UML para describir las funcionalidades que implementa el sistema. En estos diagramas observaremos de manera simplificada qué acciones ofrece el sistema dependiendo del tipo de usuario o actor en UML. Para entrar en más detalle, cada diagrama de casos de uso se complementará con un diagrama de secuencia, en el que se visualizará el flujo de acciones que realiza el sistema para ofrecer cada funcionalidad o caso de uso UML.

A continuación se detallarán las funcionalidades que ofrece el sistema, según el tipo de usuario. Se distinguen dos tipos de usuarios en el sistema, cada uno de los cuales tiene una serie de funcionalidades asociadas, los especialistas del mundo sanitario y el administrador.

### 4.2.1. Funcionalidades asociadas a los especialistas

Los especialistas encargados del tratamiento de los niños, serán los principales actores de nuestro sistema. Por ello, se les ofrecen las funciones básicas de gestión necesarias para cumplir con los requisititos mencionados. Como se ha dicho, será necesario ofrecer una gestión de las terapias que realizan y por tanto, una gestión completa de los datos de los niños que el sistema maneja. En el siguiente diagrama se detallan las funcionalidades concretas que se ofrecen a los especialistas (Figura 10).

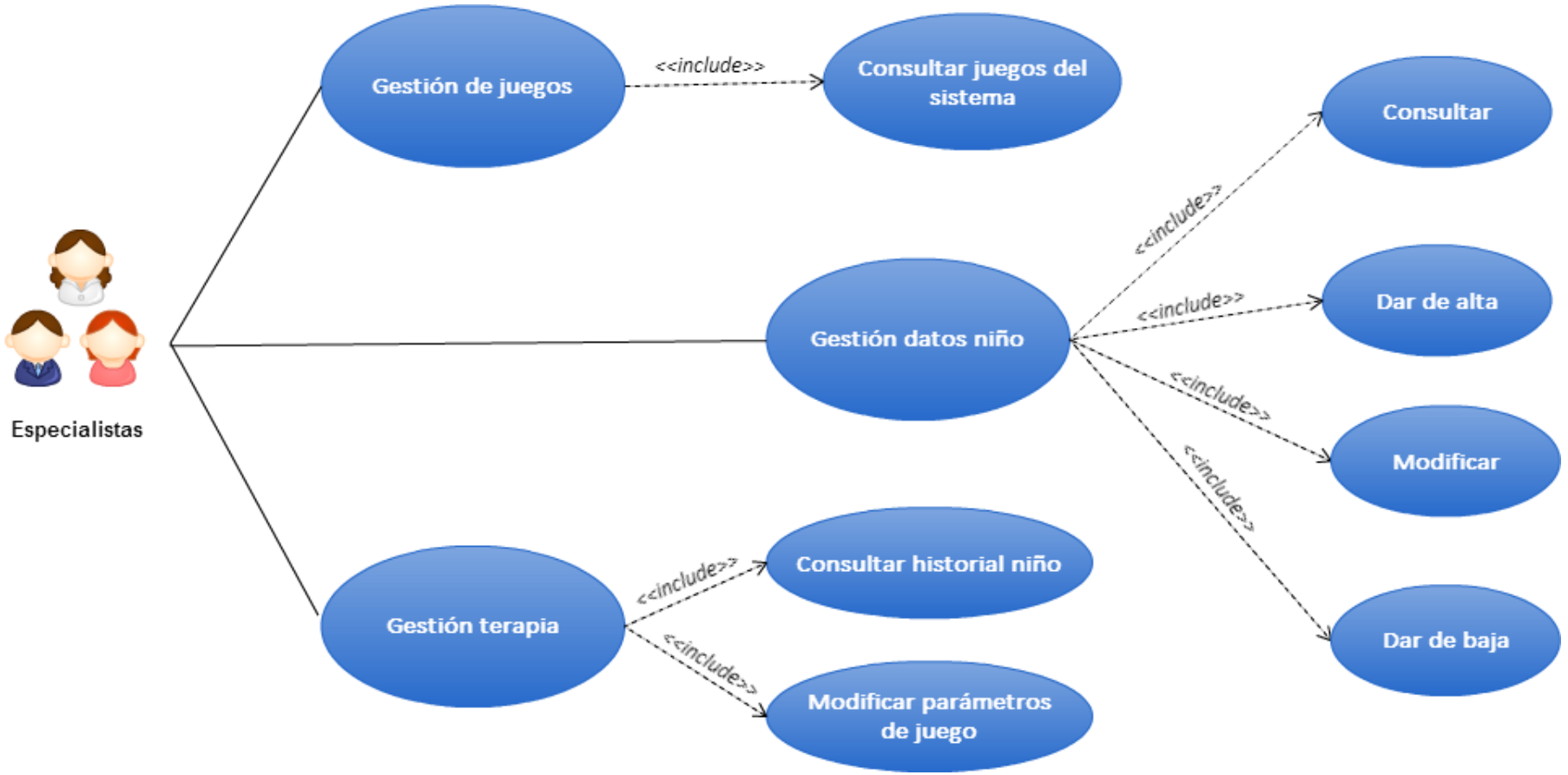


*Figura 10. Diagrama de casos de uso del sistema usuarios especialistas*

Las funcionalidades o casos de uso principales, indicados en la figura anterior, son la gestión de niños, juegos y terapias. Se analizará cada funcionalidad con mayor detalle incluyendo el diagrama de secuencia asociado a cada acción.

## Gestión datos niño

Es de vital importancia, para que el sistema sea realmente usable, posibilitar la gestión de los datos relativos a los niños involucrados en la terapia. Esta acción ha de ser realizada por el especialista encargado de cada niño, por ello, se han implementado las siguientes funciones básicas asociadas a la gestión de niños, asegurando así el cumplimiento de los objetivos del proyecto:

- **Consulta de los datos de los niños involucrados en el sistema**. Cada especialista una vez autenticado como usuario, podrá visualizar los datos del conjunto de niños a los que atiende. En la Figura 11, se muestra la secuencia de acciones que el sistema lleva a cabo para obtener y devolver los datos a la aplicación web.

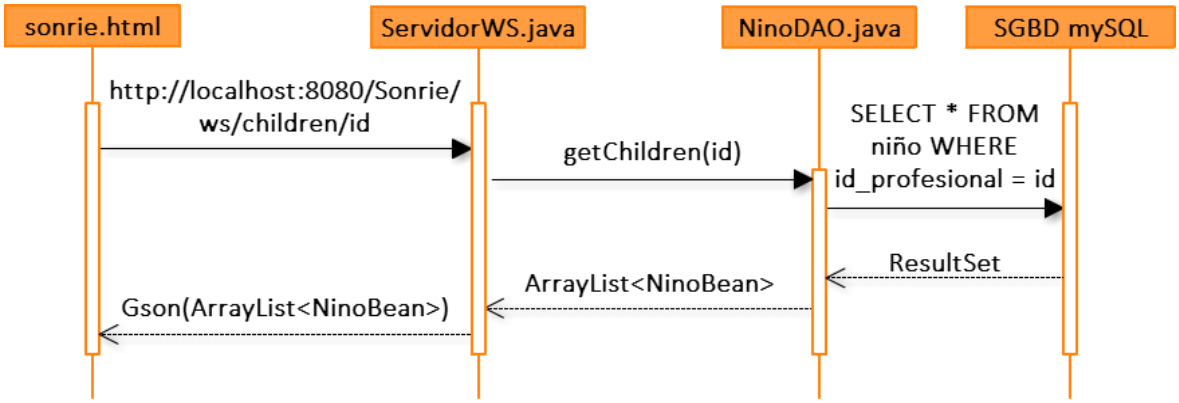


*Figura 11. Diagrama de secuencia consultar datos niño*

La aplicación web se encarga de consumir el servicio web, lanzando una petición distinta para cada acción y por tanto, para cada recurso disponible. En este caso, solicita un conjunto formado por los niños a los que atiende el especialista invocando la URI adecuada y enviando el identificador del profesional como parámetro, asegurando que cada profesional visualice una tabla únicamente con los datos de los niños a los que atiende. El servicio web instancia un objeto de tipo NinoDAO e invoca al método encargado de devolver el conjunto de niños. Internamente en todas las operaciones, en primer lugar se obtiene la conexión y posteriormente se ejecuta la sentencia SQL. Una vez se recibe el conjunto de respuestas en un objeto ResultSet, el DAO se encarga de mapear este ResultSet en un ArrayList de NinoBean, devolviendo posteriormente el resultado al servicio web y cerrando la conexión establecida. Una vez recibido, el servicio web procede a su transformación en datos legibles de manera sencilla por parte de la aplicación. Así crea un documento JSON (JavaScript Object Notation) que contiene la información recibida y lo devuelve

a la aplicación que se encarga de mostrar los datos obtenidos en una tabla seleccionable. Esta tabla permite una vez seleccionado una fila, modificar los datos del niño asociado si se pulsa el botón Modificar o dar de baja del sistema al niño seleccionado una vez se ha pulsado el botón Eliminar.

- **Dar de alta a nuevos niños en el sistema**, introduciendo los campos necesarios para identificar a un niño en el sistema correctamente. El conjunto de datos necesarios puede visualizarse en el apartado dedicado a la aplicación web. En este caso, la dirección invocada desde la web es la correspondiente al método de alta de niños, y contiene los valores que posteriormente se introducirán en la base de datos. Para ello, se sigue el esquema de funcionamiento recogido en el diagrama de secuencia que aparece a continuación :

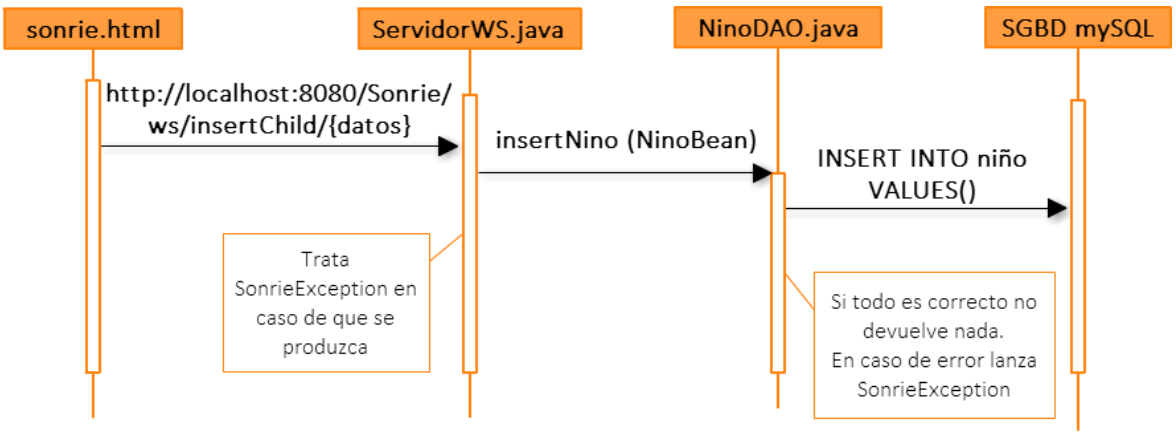


*Figura 12. Diagrama de secuencia dar de alta niño*

En el caso anterior, así como en el resto de casos, primero se solicita la conexión. Una vez recibida se ejecuta la sentencia INSERT con los valores recibidos y se procede a cerrar la conexión. No se ha considerado necesario devolver ningún valor pues, en caso de producirse algún error el servicio web recibirá la excepción e informará de ello a la aplicación que mostrará el mensaje correspondiente.

- **Modificación de los datos de los niños dados de alta en el sistema**. Esta funcionalidad permite, una vez seleccionado un niño en la tabla de consulta de niños de cada profesional, ofrecer un formulario con los valores de los campos del registro del niño en cuestión de forma que, el profesional pueda modificarlos en caso necesario (ver Figura 13).

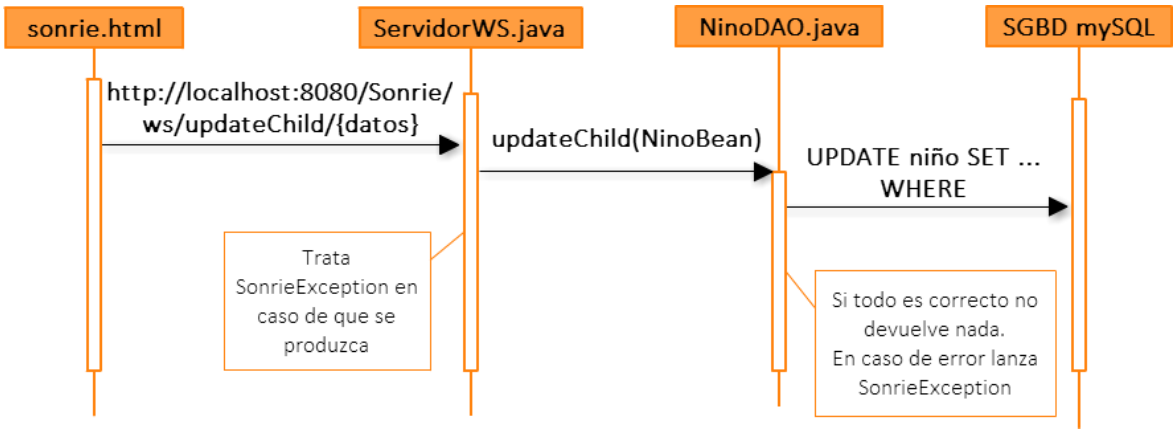


*Figura 13. Diagrama de secuencia modificar datos niño*

Para que esta modificación sea correcta, la cláusula WHERE de la sentencia SQL ejecutada desde el DAO, incluye una restricción por la clave primaria de la tabla niño, es decir, el identificador del niño en la tabla. De esta forma, aseguramos que estamos modificando el registro correcto. Esta sentencia devuelve un error que se transforma en una excepción de tipo SonrieException, si el profesional encargado del niño no existe en el sistema previamente. Este error es manejado por el servicio web informando a la aplicación. En caso de un funcionamiento correcto, se actualiza el registro correspondiente de la tabla que contiene los niños dados de alta en el sistema cuyo profesional encargado del proceso de terapia es el autenticado en la sesión actual de la aplicación.

- **Dar de baja a niños del sistema**. Esta funcionalidad permite borrar del sistema los datos del niño seleccionado en la tabla de consulta de niños. El proceso que se produce a nivel interno se detalla a continuación:

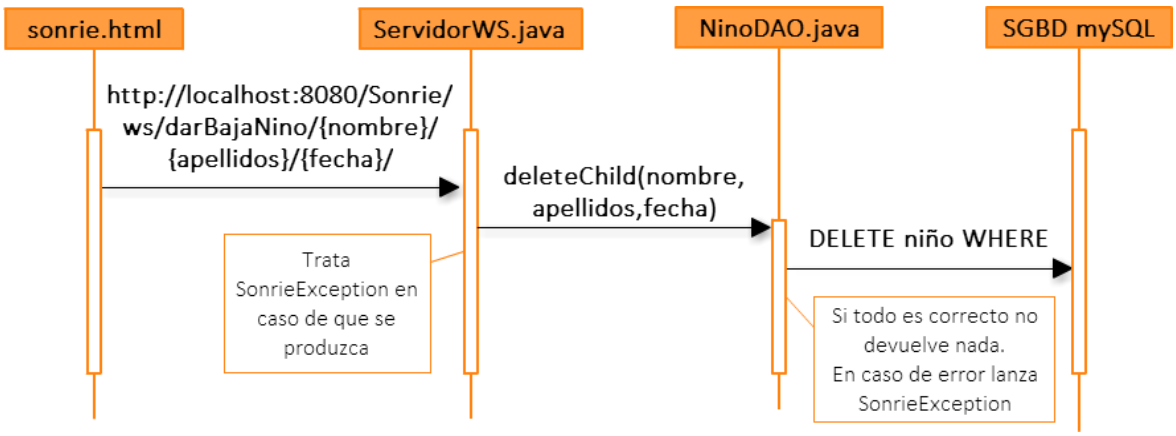


*Figura 14. Diagrama de secuencia dar de baja niño*

Cabe destacar las siguientes consideraciones:

- En primer lugar, la cláusula WHERE de la sentencia que ejecuta el componente DAO, incluye la restricción de los campos que se envían como atributos, estos son: nombre y apellidos del niño y su fecha de nacimiento. Es decir, se eliminará del sistema el registro que contenga dichos valores en sus campos: nombre, apellidos y fecha.
- En este caso, dado que partimos de los datos de la tabla consulta, no existe posibilidad de errores en cuanto a restricciones de claves, sin embargo, podría ocurrir algún error con la conexión. En este caso, es el servicio web el encargado del tratamiento de la excepción y la devolución del correspondiente mensaje a la aplicación que lo mostrará por pantalla. En una ejecución normal, la tabla de datos de los niños se actualiza, con la desaparición del registro correspondiente.

## Gestión de juegos

Los especialistas podrán consultar los juegos implementados en la plataforma, se mostrará el nombre del juego y el objetivo. Para ello el sistema lleva acabo la siguiente secuencia de acciones.

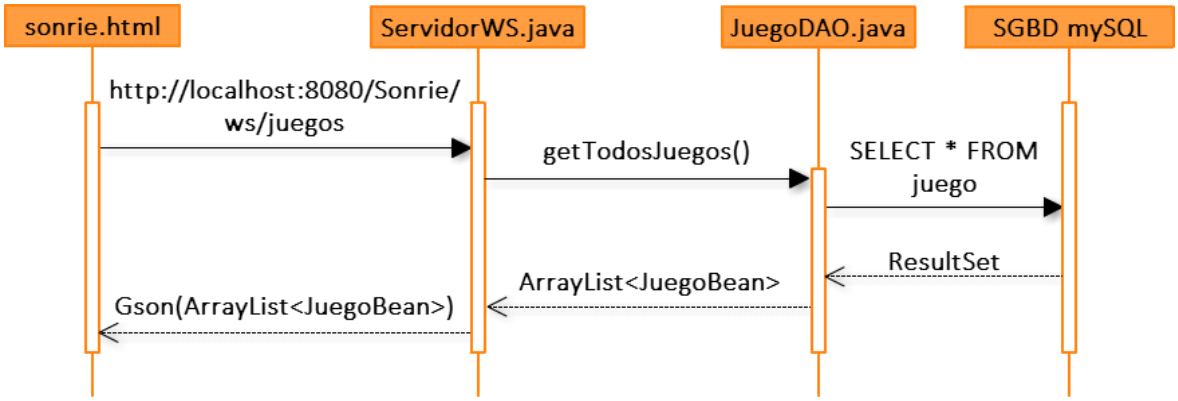


*Figura 15. Diagrama de secuencia consultar juegos del sistema*

La aplicación web solicita un conjunto formado por los juegos del sistema invocando la URI de la Figura 15. El servicio web instancia un objeto de tipo JuegoDAO e invoca al método encargado de devolver el conjunto de juegos. Internamente, como en el resto de operaciones, se obtiene en primer lugar la conexión y, posteriormente se ejecuta la sentencia SQL. Una vez se recibe el conjunto de respuestas en un objeto ResultSet, el DAO se encarga de mapear este ResultSet en un ArrayList de JuegoBean, devolviendo posteriormente el resultado al servicio web, cerrando previamente la conexión establecida con la BD. Posteriormente, el servicio web procede a su transformación en datos legibles de manera sencilla por parte de la aplicación, en un documento JSON que contiene la información recibida. Desde el lado del cliente, una vez ha recibido este documento, se visualizan los datos obtenidos en forma de tabla. Esta tabla no es seleccionable puesto que las funciones de modificación, alta y baja solo se ofrecen al usuario administrador.

## Gestión de terapia

Esta funcionalidad está destinada a ofrecer el objetivo principal, una terapia efectiva. Para ello se han implementado dos funcionalidades básicas para los especialistas médicos.

- **Consulta de historial**. En este menú los especialistas podrán consultar el historial de juego de cada niño en la plataforma SONRIE. Para ello, se mostrará una tabla en la que se incluyen datos de interés para los profesionales. Para cada intento y juego realizado, los datos de interés que el sistema almacenará son el tiempo empleado y el uso de otros músculos, que no intervendrían en la ejecución normal del movimiento. En el siguiente diagrama de secuencia pueden observarse el flujo de acciones necesarias para ofrecer estos datos a los especialistas.

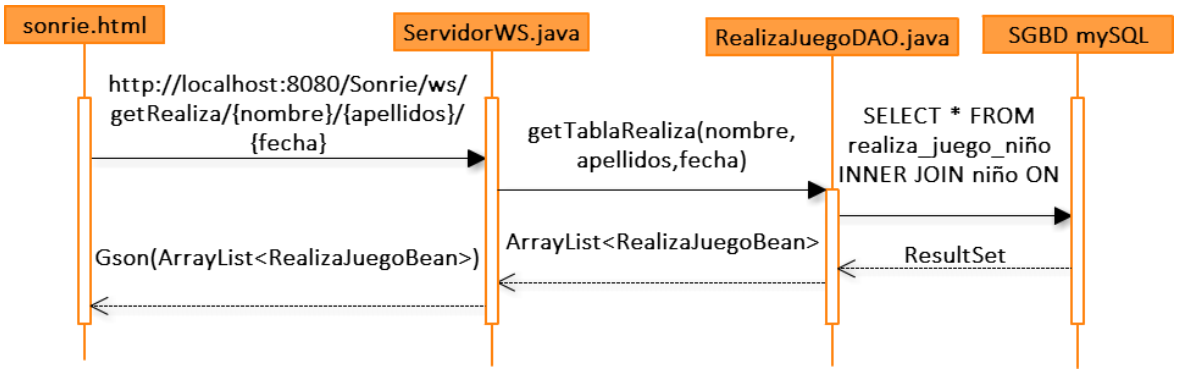


*Figura 16. Diagrama de secuencia consultar historial niño*

- **Modificación de parámetros de juego**. En concreto, el número de repeticiones y el tiempo límite por intento. El objetivo de esta tarea, es introducir en el sistema modificaciones en cuanto a la dinámica de los juegos para cada niño. Esta dinámica queda determinada por el número de repeticiones de cada ejercicio, así como, por el tiempo límite disponible para cada una de estas repeticiones. Por tanto, este menú permite seleccionar un niño que recibe terapia con el profesional que ha iniciado la sesión en el sistema y modificar estos datos, de manera que la próxima vez que el sistema se ejecute será con estos parámetros. El objetivo principal es la adaptación completa del sistema al tratamiento del niño. Esto significa que, en caso de mejora en algún movimiento o en caso de que los valores por defecto no sean suficientes para conseguir una terapia efectiva el sistema se adaptará en función de los criterios de los especialistas introducidos a través del menú de modificación dentro de la opción de Gestión de terapia. Además el sistema, muestra en primer lugar una tabla con todas las configuraciones introducidas para un niño en concreto utilizando las acciones descritas en el diagrama de secuencia de la Figura 17.

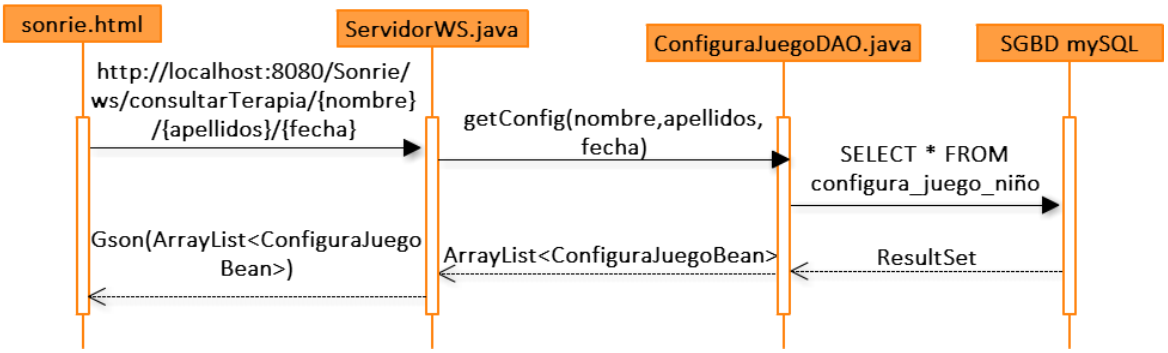


*Figura 17. Diagrama de secuencia consultar configuraciones de un niño*

Una vez visualizadas las configuraciones de los juegos para un niño en concreto, el sistema permite a los profesionales añadir una nueva configuración para un juego determinado. Para ello debe introducir los datos que determinan una nueva configuración, estos son: el nombre del juego, nombre y apellidos del niño, la fecha en la que se introduce la configuración y un conjunto de datos que condicionarán el funcionamiento del juego indicado, y que son: el tiempo límite que se permite para ejecutar cada intento de este juego, así como, el número de intentos o repeticiones que se consideran necesarias para este niño. El sistema almacena un valor por defecto de tres intentos y 15 segundos por intento, sin embargo, estos datos podrán modificarse añadiendo una nueva configuración. Así, desde la aplicación web se recogerán los datos juego, nombre, apellidos, fecha, tiempo y repeticiones, y se enviarán en la invocación de la URL del servicio, como puede verse en la Figura 18. El flujo de acciones llevado a cabo por el sistema para añadir esta nueva configuración a la base de datos con el fin de cargarla posteriormente en el sistema de juegos queda descrito en el siguiente diagrama de secuencia:

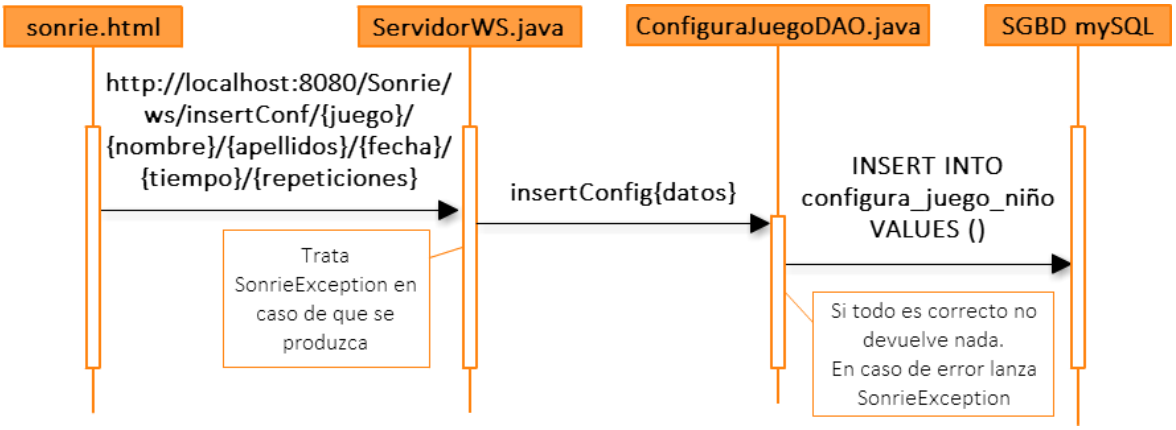


*Figura 18. Diagrama de secuencia añadir configuración niño y juego*

### 4.2.2. Funcionalidades asociadas al administrador del sistema

Será necesario un usuario administrador del sistema que dé de alta y de baja a usuarios en el sistema y que tenga acceso a sus datos, así como, a su modificación. Para ello, en primer lugar deberá autenticarse en el sistema. Una vez validada la clave, además de funciones de gestión de profesionales, tendrá acceso a funciones relativas a una edición activa de los juegos detallados en la base de datos y, por tanto, pertenecientes a la plataforma de juegos. Estas funciones activas se refieren a la modificación, alta y baja de juegos. Obviamente dispone de la opción de consulta de juegos del sistema. A continuación, se muestran los diagramas asociados a cada caso de uso comentado y sus funcionalidades asociadas.

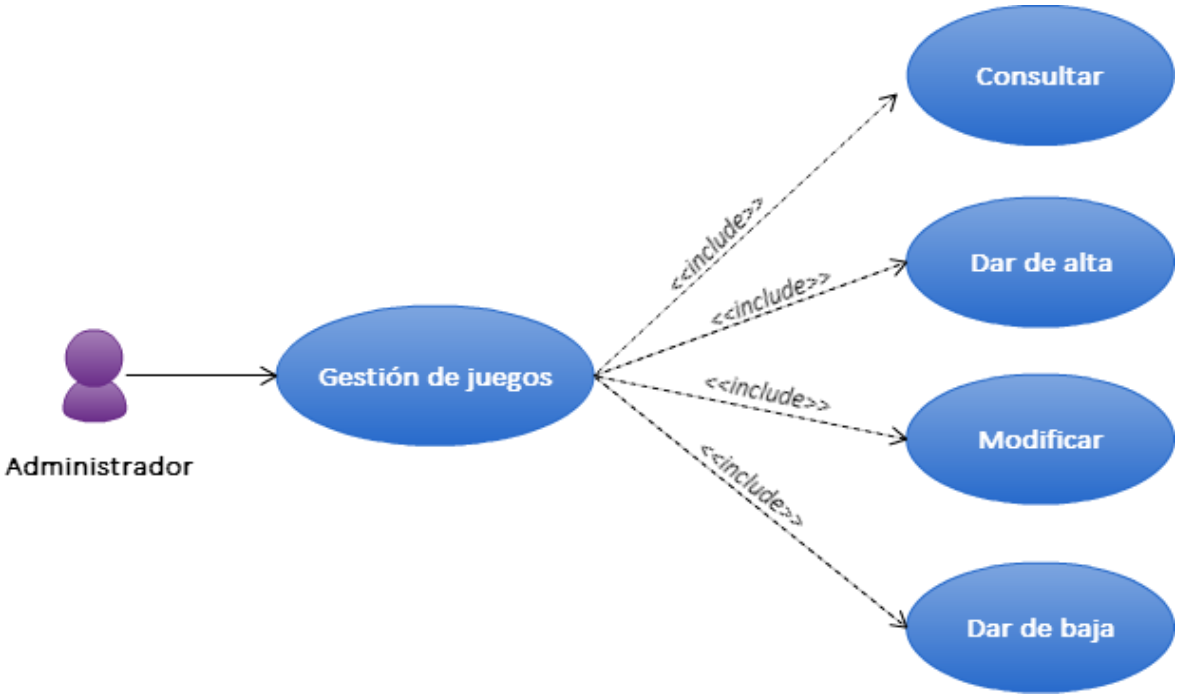


*Figura 19. Diagrama de caso de uso gestión de juegos del administrador*

La secuencia de acciones necesarias para ofrecer estas funcionalidades es equivalente a las anteriores, utilizando un objeto JuegoDAO como objeto encargado del acceso a los datos. De forma análoga, se utilizará un objeto de tipo ProfesionalDAO para llevar a cabo las siguientes funciones asociadas a la gestión de especialistas:

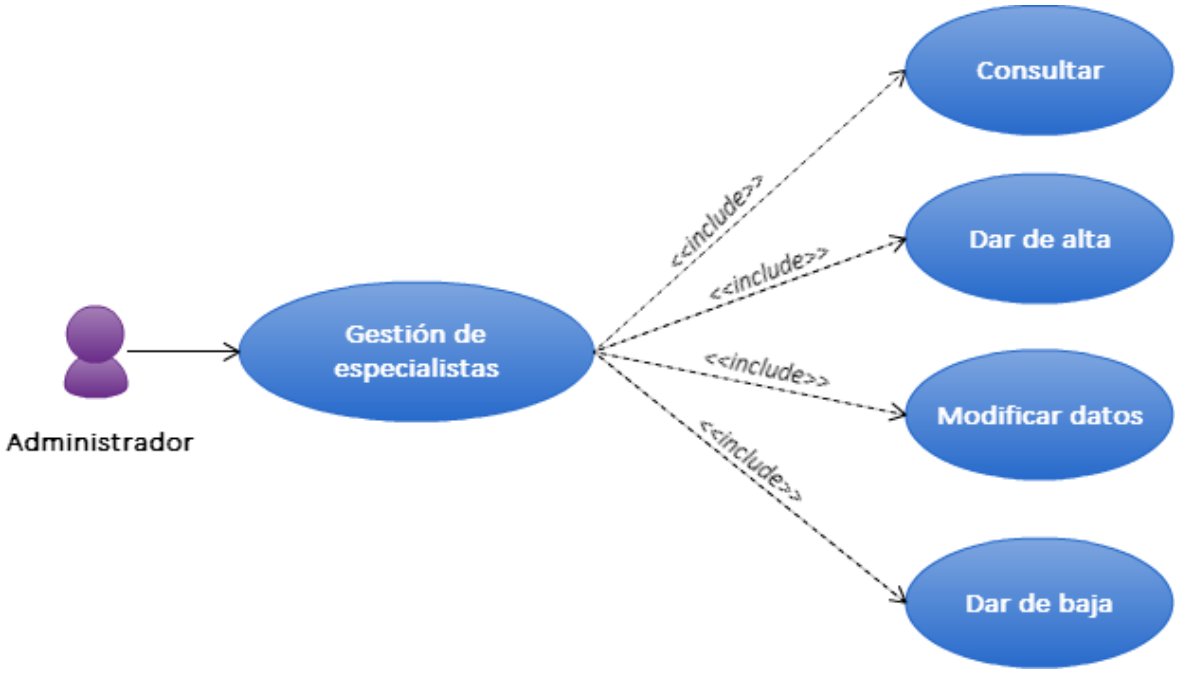


*Figura 20. Diagrama de caso de uso gestión de especialistas*

Estas funcionalidades únicamente se ofrecen si la contraseña introducida es válida. Para asegurar seguridad en el acceso a la aplicación, el sistema almacena el hash de la clave del administrador en un fichero en lugar de la propia clave. De esta forma, cuando se introduce la contraseña, a través del menú de autenticación de la aplicación web, primeramente se calcula su hash y posteriormente se compara este valor hash generado con el valor hash almacenado en el sistema. Con lo que conseguimos aportar seguridad en el acceso al sistema, ya que es imposible obtener la clave del administrador a partir del hash almacenado.

Cabe destacar el mecanismo llevado a cabo para dar de alta a un especialista, pues se diferencia del resto de inserciones del sistema, ya que un profesional tiene asociado, además del registro de la tabla profesional, un registro en la tabla usernames y otro en la tabla passwords para poder autenticarse en el sistema. Por tanto, una vez invocada la URL y construido el DAO que nos proporciona la conexión a la base de datos, se realizan **tres inserciones** en el mismo método.

Por defecto, una conexión se crea en modo auto-commit, esto significa, que cada sentencia SQL se trata como una transacción individual y se compromete de forma automática justo después de completarse la ejecución de la sentencia. Comprometer una transacción significaría, en nuestro caso, que los datos de la primera transacción (que corresponde a la primera inserción) se almacenarían en la base de datos, y si la segunda transacción fallase, el SGBD no podría restaurar el estado anterior de la BD. No tendría sentido y podría ocasionar errores en el sistema, por ejemplo, registrar un usuario si el propio profesional no está registrado en el sistema, o registrar la contraseña si el usuario no existe.

Por este motivo, en el caso del alta de un especialista, debemos agrupar las tres sentencias y sus ejecuciones en la misma transacción asegurando la consistencia de la información del sistema. De esta forma, aseguramos que si alguna de las sentencias produce algún error, la transacción completa se deshace y por tanto, no se ejecuta ninguna de las tres sentencias y la información del sistema continúa siendo consistente.

El método insertProfesional() de ProfesionalDAO se encarga de desactivando el modo auto-commit antes de ejecutar el flujo de acciones que puede verse en la siguiente figura. Una vez que se ha dado de alta al nuevo profesional con un nombre de usuario y contraseña asociados, se reactiva el modo auto-commit y se cierra la conexión con la base de datos.

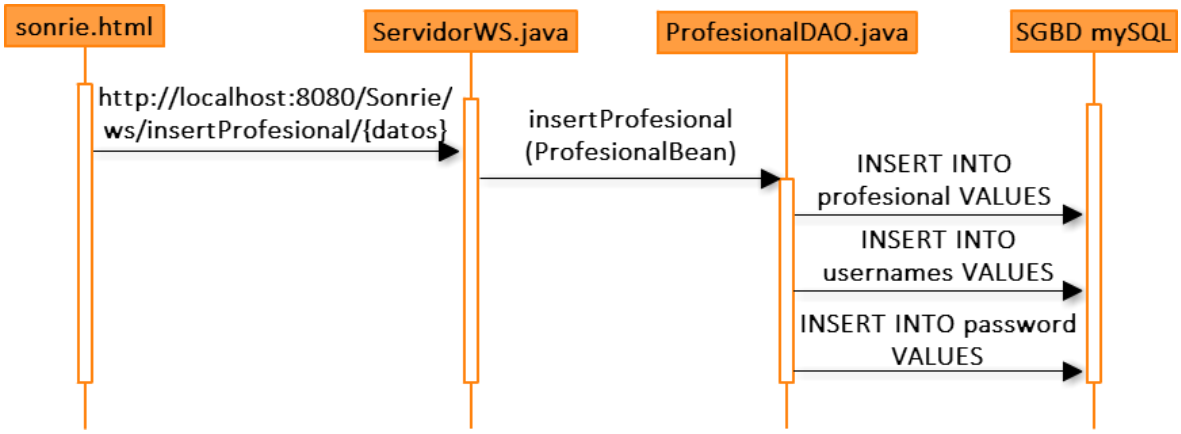


*Figura 21. Diagrama de secuencia dar de alta profesional*

Lógicamente, este mismo mecanismo se utiliza para dar de baja al profesional del sistema, eliminando los registros asociados a las tablas profesional, usernames y passwords del sistema en la misma transacción.

El sistema permite la modificación de los pares clave-valor, así como del resto de datos del profesional, en la opción modificar datos. Sin embargo, el acceso a esta funcionalidad está limitado únicamente al administrador del sistema.

## 4.3.Base de datos

Debido a los requisitos del sistema será necesario almacenar algunos datos para poder ofrecer los servicios mencionados. Para ello y como soporte fundamental de nuestra plataforma, se ha implementado una base de datos dentro de un Sistema Gestor de Base de Datos Relacional (MySQL).

La función principal de la base de datos MySQL es asegurar la persistencia de los datos de la aplicación desarrollada. Por su parte, el sistema gestor de la base de datos nos proporciona las funciones básicas necesarias para añadir, modificar, borrar y consultar los datos.

Con el fin de ofrecer los servicios descritos, será necesario definir una serie de entidades correspondientes a los principales elementos que intervienen en nuestro sistema. Estas entidades se traducen, en última instancia, en tablas de datos para la BD MySQL que comentaremos más adelante. La primera cuestión que nos surge a la hora de diseñar el sistema es la selección y modelado de las distintas entidades así como las relaciones que han de existir entre ellas. Las entidades fundamentales que necesita nuestro sistema para cumplir con las especificaciones descritas son, los **niños**, los **profesionales** y los **juegos** del sistema. Además, será necesario definir otras entidades como el **centro** de trabajo de los profesionales o su **especialidad** para construir un sistema completo. Así, se ha diseñado el siguiente modelo entidad-relación del sistema (Figura 22) en el que se pueden observar las entidades que intervienen en el sistema así como, las relaciones que existen entre ellas.

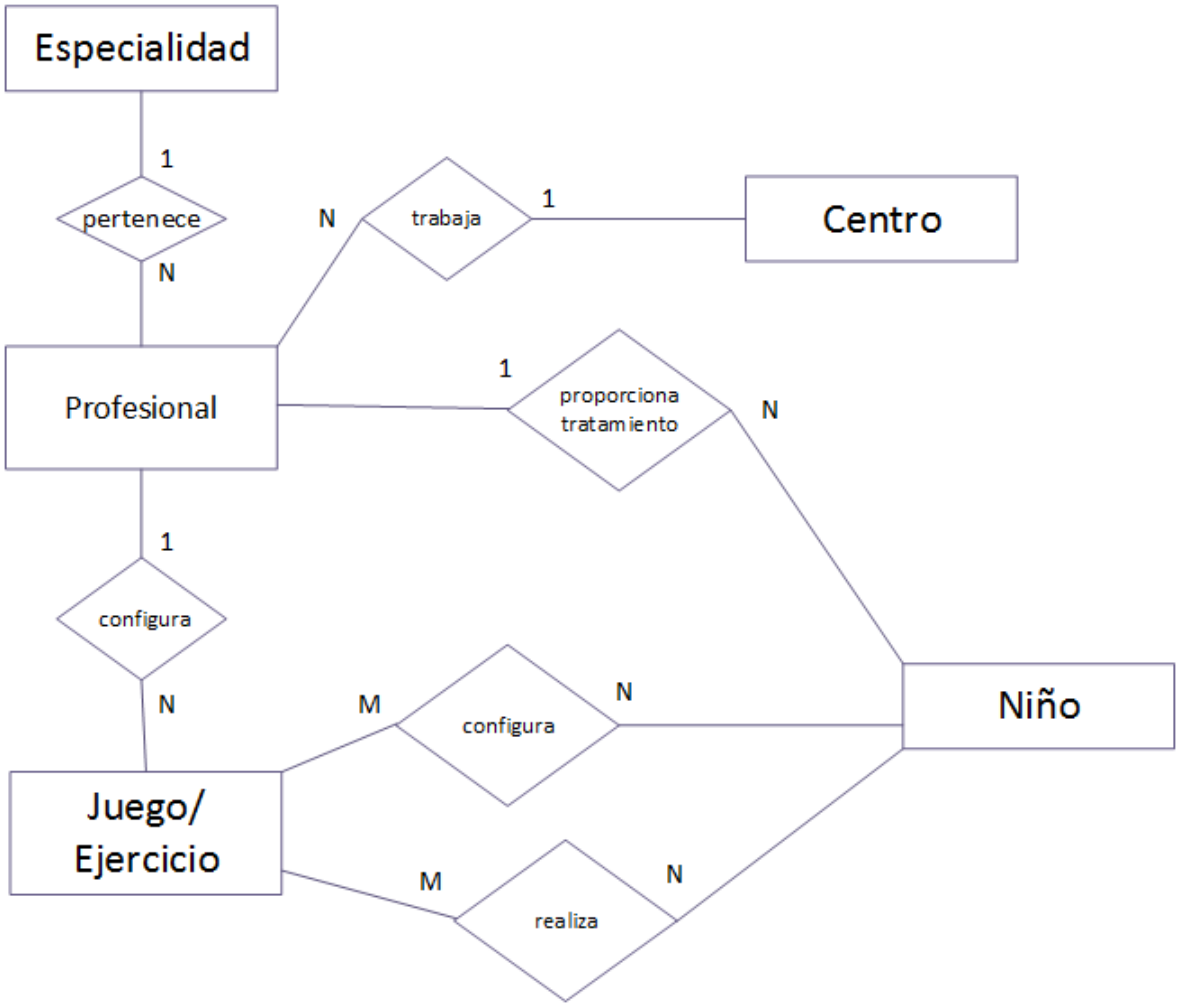


*Figura 22. Diagrama Entidad-Relación Base de Datos Sonríe*

Como puede observarse en la figura anterior, cada profesional que utilice el sistema SONRIE, proporcionará tratamiento a varios niños, los cuales realizarán el conjunto de ejercicios definidos en forma de juegos. Los profesionales podrán configurar los juegos de los niños a los que proporcionen terapia de manera que cada niño tendrá su configuración para ese juego ajustándose a sus necesidades. Se han definido las entidades centro y especialidad para albergar la información relativa al centro de trabajo del profesional y el campo en el que se han especializado, respectivamente.

Las entidades representadas corresponden directamente a tablas SQL que se deberán incluir en la base de datos. Además, algunas relaciones, condicionan la creación de tablas simplificando así el tratamiento de la información y por tanto la lógica del sistema. Podemos observar la relación que nos informa que varios niños realizan varios ejercicios del sistema. La realización de cada juego por parte de cada niño ha de ser almacenada si queremos ofrecer un historial de consulta a los profesionales. Por tanto se ha diseñado e implementado una tabla denominada **realiza_juego_niño** para conseguir este objetivo. Análogamente, existe la relación de configuración de los juegos para cada niño. Obviamente los datos de cada configuración deberán ser persistentes para que el sistema

sea capaz de usarlos para ofrecer la terapia a los niños. Así, se ha construido la tabla **configura_juego_niño**, con esta misión fundamental. Estas tablas se explicarán con mayor detalle a continuación.

Partiendo del diagrama entidad-relación se ha decidido definir el siguiente modelo relacional (Figura 23). En él podemos visualizar las tablas que componen el sistema y sus relaciones. El conjunto de tablas definidas, albergará toda la información necesaria para asegurar la persistencia del sistema y su dinamismo.

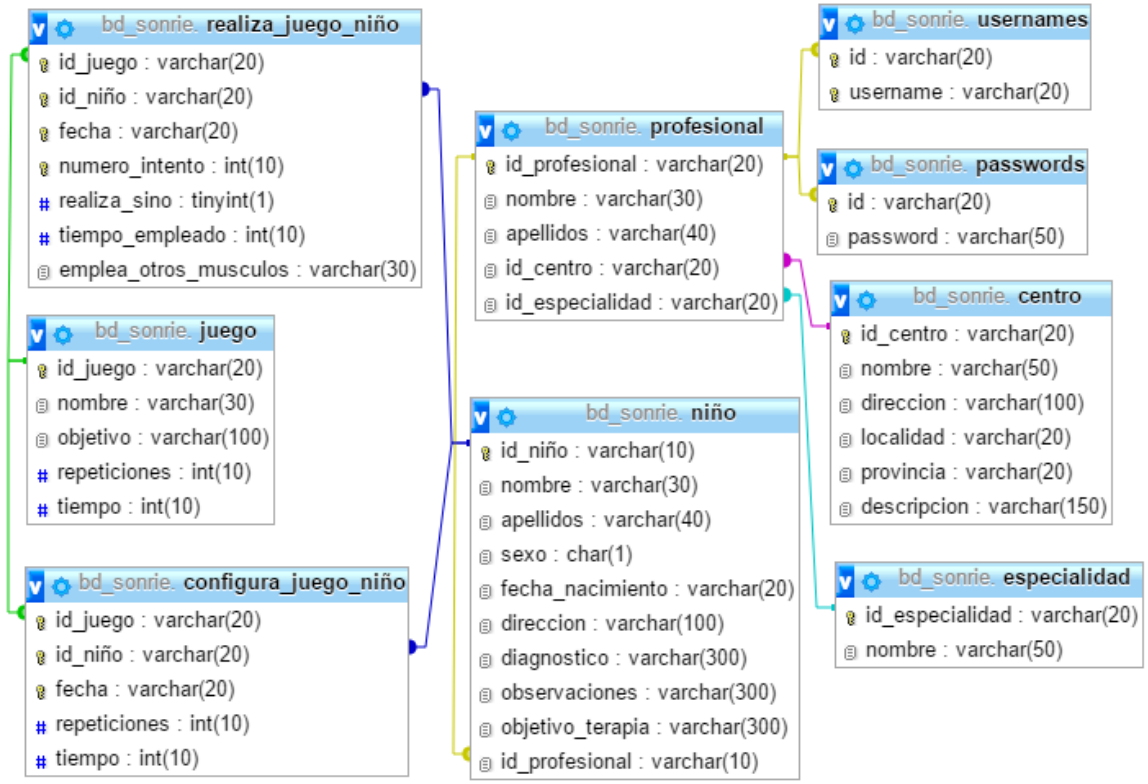


*Figura 23. Modelo relacional*

A continuación, se hará un recorrido de las tablas definidas en la base de datos, representadas en la figura anterior.

### 4.3.1. Información persistente de juegos, niños y profesionales

Para garantizar la persistencia del sistema en primer lugar será necesario almacenar la información asociada a los niños y a los profesionales que utilizan SONRIE, así como los datos de los juegos del sistema, permitiendo así, ofrecer las funcionalidades comentadas en el apartado anterior.

Asociado a cada niño que utilice el sistema se almacenará una entrada en la **tabla niño** que contendrá los datos de cada niño que utilice el sistema:

*Tabla 1. Tabla de niños*

| Niño | |
|---|---|
| (PK) | Id_niño : varchar(10) |
| | Nombre : varchar(30) |
| | Apellidos : varchar(40) |
| | Sexo : varchar(1) |
| | Fecha_nacimiento : varchar(20) |
| | Direccion : varchar(100) |
| | Diagnostico : varchar(300) |
| | Observaciones : varchar(300) |
| | Objetivo_terapia : varchar(300) |
| (FK) | Id_profesional: varchar(10) |

De estos atributos destacaremos los siguientes:

- El identificador del niño ha de ser único, se recomienda utilizar el DNI.
- El campo dirección del niño es opcional.
- Diagnóstico del niño. Un niño diagnosticado con PCI puede tener diagnosticados otras enfermedades, trastornos, déficits, etc.
- Observaciones a tener en cuenta.
- Objetivos que se pretenden conseguir con la terapia.
- Id del profesional encargado de la terapia del niño.

La tabla profesional por su parte contendrá los siguientes datos de cada profesional que utilice el sistema SONRIE:

*Tabla 2. Tabla de profesionales médicos*

| Profesional | |
|---|---|
| (PK) | Id_profesional : varchar(10) |
| | Nombre : varchar(30) |
| | Apellidos : varchar(40) |
| (FK) | Id_centro : varchar(20) |
| (FK) | Id_especialidad : varchar(20) |

Destacan los siguientes atributos de la tabla profesional:

- Identificador único del profesional (Clave primaria). Se recomienda utilizar el DNI.

- Id del centro al que pertenece el profesional (Clave extranjera que referencia a la entrada correspondiente al profesional en la tabla centro)
- Id de la especialidad del profesional (Clave extranjera que referencia a la entrada correspondiente a este profesional en la tabla especialidad)

La tabla juego contendrá los datos asociados a cada juego del sistema.

*Tabla 3. Tabla de juegos*

| Juego | |
|---|---|
| (PK) | Id_juego: varchar(20) |
| | Nombre : varchar(30) |
| | Objetivo : varchar(100) |
| | Repeticiones : int(10) |
| | Tiempo : int(10) |

Cabe destacar los siguientes campos:

- Identificador único del juego (Clave primaria)
- Objetivo terapéutico que se pretende con la realización del juego. Irá siempre encaminado a amortiguar la PCI ya diagnosticada y a la mejora de su calidad de vida, facilitando en el niño el proceso de alimentación, aseo, comunicación, entre otros.
- Número de repeticiones del juego. Este campo es de vital importancia para conseguir el fin terapéutico deseado y para asegurar una aceptación positiva de SONRIE. El sistema ofrece la posibilidad de aumentar las repeticiones de los juegos conforme avanza la terapia y el grado de aceptación del niño. Es fundamental que el niño vea el sistema como un entorno lúdico, por tanto, en las primeras ejecuciones de los ejercicios de SONRIE, debemos asegurar que este número de repeticiones resulta motivante para el niño, en lugar de crearle una situación de estrés o frustración. A medida que el niño avance en la ejecución de los movimientos se podrá aumentar o disminuir este parámetro en función del criterio del profesional encargado de su tratamiento.
- Tiempo máximo disponible para cada repetición de este juego.

### 4.3.2. Información a almacenar para datos de usuarios médicos

Asociadas a la tabla profesional existen, en primer lugar, un par de tablas destinadas a almacenar la información sobre el centro y la especialidad del profesional. En concreto:

*Tabla 4. Tabla de centros*

| Centro | |
|---|---|
| (PK) | Id_centro: varchar(20) |
| | Nombre : varchar(50) |
| | Dirección : varchar(100) |
| | Localidad : varchar(20) |
| | Provincia: varchar(20) |
| | Descripción : varchar(150) |

*Tabla 5. Tabla de especialidades*

| Especialidad | |
|---|---|
| (PK) | Id_especialidad: varchar(20) |
| | Nombre : varchar(50) |

Además para poder utilizar el sistema será necesario un nombre de usuario y una clave asociada. Para ello se han definido las siguientes tablas, que almacenan los datos fundamentales necesarios para la autenticación:

*Tabla 6. Tabla de usernames*

| Usernames | |
|---|---|
| (FK) | Id: varchar(20) |
| | Username: varchar(20) |

En la tabla usernames se almacenará el ID del profesional asociado a un username con el que tendrá acceso a la aplicación.

*Tabla 7. Tabla de passwords*

| Passwords | |
|---|---|
| (FK) | Id: varchar(20) |
| | Password : varchar(50) |

Como se comentó anteriormente, en los casos de uso del administrador del sistema, se pretende dotar de seguridad en la autenticación a la aplicación web. Por ello, en la tabla passwords se almacenará el ID del profesional asociado al hash de su clave de acceso al sistema. Con esta separación de los datos básicos que intervienen en la autenticación, username y password, conseguimos dotar de seguridad a nuestro sistema. Esta seguridad en el acceso se obtiene, en primer lugar, separando el par usuario-clave en tablas distintas, y, en segundo lugar, almacenando el hash de la clave en vez de la propia clave. Es imposible obtener la clave del usuario a partir del hash, con lo que limitamos el acceso únicamente a usuarios del sistema.

### 4.3.3. Información necesaria para realizar una terapia efectiva

La misión fundamental de esta plataforma es ofrecer una terapia efectiva a los niños dados de alta en el sistema. Por ello, se ofrece la posibilidad a los especialistas de configurar los juegos en función de las necesidades y progresos de cada niño. De este hecho deriva la creación de las siguientes tablas, necesarias para asegurar que las configuraciones elegidas por los profesionales (Tabla 8) y los resultados de la ejecución de los juegos por parte de cada niño (Tabla 9), persistan en el sistema. De esta forma, cuando la plataforma de juegos inicie su ejecución, se cargarán los valores almacenados asociados a cada niño, ofreciendo una dinámica de juego adecuada, y, cuando termine, los resultados obtenidos se almacenarán para su posterior revisión por parte de los especialistas.

*Tabla 8. Tabla configura juego niño*

| | Configura juego niño | |
|---|---|---|
| (FK) | Id_juego: varchar(20) | PK |
| (FK) | Id_niño : varchar(20) | |
| | Fecha : varchar(20) | |
| | Repeticiones : int(10) | |
| | Tiempo : int(10) | |

En la tabla **configura juego niño** destacamos los siguientes atributos:

- El conjunto de claves extranjeras formado por el identificador del juego, el identificador del niño, junto con la fecha, constituyen la clave primaria. De esta forma se dispondrá de un historial de las configuraciones realizadas para cada juego de cada niño.
- Número de repeticiones a realizar de cada juego. Este parámetro por defecto tiene valor 3, pero se ofrece la posibilidad de realizar cada un juego un número superior o inferior de veces si el especialista lo requiere.
- Tiempo máximo permitido en cada intento. Este valor es de 15 segundos. Sin embargo, el especialista puede modificarlo en caso de ser necesario para ofrecer una terapia más adecuada a cada niño.

*Tabla 9.Tabla realiza juego niño*

| Realiza juego niño | | |
|---|---|---|
| (FK) | Id_juego: varchar(20) | PK |
| (FK) | Id_niño : varchar(20) | |
| (FK) | Fecha : varchar(20) | |
| | Número_intento : int | |
| | Realiza_si_no : varchar(10) | |
| | Tiempo_empleado : int(10) | |
| | Emplea_otros_musculos: varchar(30) | |

La tabla **realiza juego niño**, contiene los detalles de los resultados de la ejecución de cada intento de cada juego por parte de cada niño. Por ello,

- El conjunto de atributos que constituyen la clave primaria de la tabla serán el identificador del juego, el identificador del niño, la fecha y el número del intento del que se trate.
- Para el conjunto de datos que forman la clave primaria, se almacenará un indicador de la realización positiva o negativa del ejercicio, el tiempo empleado para ejecutarlo y, un campo que indicará si se han utilizado otros músculos a la hora de ejecutar el movimiento. Este último campo es un requisito de vital importancia para los especialistas, pues necesitan conocer si el niño está ejercitando los músculos encargados de los gestos implementados o, si se está compensando en movimiento del músculo que presenta dificultades de movimiento, con el movimiento de otros músculos, algo que ocurre muchas veces en los niños con PCI.

## 4.4. Diseño e implementación del acceso a datos

La capa de acceso a datos, como ya se ha comentado, se encuentra entre la lógica de negocio y la capa de datos del sistema. Su misión fundamental es la de abstraer a las capas superiores de los mecanismos utilizados para garantizar la persistencia. Para conseguir este objetivo, se han utilizado un conjunto de clases Java encargadas del acceso a la base de datos del sistema y de la representación de los datos en forma de objetos Java. Esto último nos permite simplificar enormemente el tratamiento de los datos en capas superiores ya que los datos serán manejados como clases Java compuestas de sus atributos y métodos, relacionados con las funcionalidades del sistema detalladas anteriormente.

Para desarrollar estos componentes encargados de la abstracción de la capa de datos del sistema, se han utilizado dos tipos de patrones Java: los objetos de transferencia de datos (DTO) y los objetos de acceso a datos (DAO).

Para poder ofrecer los servicios mencionados, se necesita enviar la información contenida en la BD cuando el cliente lo demanda, así como, modificar los registros de las tablas con los datos que se reciben desde el cliente. Para llevar a cabo este proceso, se utilizarán objetos Java que representarán un registro de una tabla concreta de la BD. Esto significa que el objeto dispondrá de una serie de atributos correspondientes a los campos de la tabla de la BD que represente. Estos atributos contendrán el valor de los campos del registro concreto de la BD. Para implementar estos objetos se ha utilizado el patrón DTO. Estos DTO, solo almacenarán la información que se pretende transmitir al nivel de aplicación o que se necesita recibir de este nivel, por tanto, una vez definido el modelo relacional de la base de datos, será necesario analizar qué tablas requieren una correspondencia directa a clases Java. Para cada una de las tablas seleccionadas, se ha implementado un objeto DTO asociado, que representará un registro de la tabla y un objeto DAO, encargado de la conexión con la base de datos, la ejecución de sentencias SQL y la traducción de los datos recibidos en una instancia del DTO adecuado. Los objetos DAO utilizan los DTO para realizar operaciones sobre la BD, estableciendo previamente una conexión. Por tanto, contienen toda la lógica de la capa de acceso, mientras que los DTOS, son meros contenedores de la información del sistema que los objetos DAO utilizan a través de sus métodos.

### 4.4.1. Objetos de transferencia de datos

La implementación de los DTO se ha basado en JavaBeans, lo que supone el uso de componentes software reutilizables y modificables, encargados de encapsular los datos de la BD que el sistema maneja para ofrecer sus servicios. Cada instancia de un objeto JavaBean representará una tupla o registro de una tabla concreta, por tanto, estará compuesta de tantos atributos como campos contenga la tabla en cuestión. Los objetos de acceso a datos se encargarán de dotar de contenido estos atributos de los objetos de transferencia de datos en función de los datos almacenados en la base de datos. La

correspondencia entre los tipos de datos MySQL definidos y los tipos de datos empleados a la hora de definir los objetos JavaBeans, se ha establecido según el mapeo de tipos de datos definido en la API JDBC.

Así cada uno de nuestros objetos JavaBeans estará compuesto por atributos privados asociados a los campos de la tabla que representa. Se han implementado los objetos de transferencia de datos detallados en la tabla 10. En la que además se muestra la tabla SQL a la que hace referencia cada clase Java.

*Tabla 10. Objetos de transferencia de datos*

| Objetos de transferencia de datos | Tabla representada |
|---|---|
| bdg.datos.NinoBean | Niño |
| bdg.datos.JuegoBean | Juego |
| bdg.datos.ProfesionalBean | Profesional |
| bdg.datos.CentroBean | Centro |
| bdg.datos.EspecialidadBean | Especialidad |
| bdg.datos.RealizaJuegoBean | Realiza_juego_niño |
| bdg.datos.ConfiguraJuegoBean | Configura_juego_niño |

Como podemos observar, las únicas tablas que no tienen una correspondencia directa con una clase Java son las destinadas a la autenticación. En la etapa de diseño se decidió definir los campos de las tablas usernames y passwords como atributos de la clase ProfesionalBean.

Las fortalezas del estándar de codificación JavaBeans, residen en sus requisitos de implementación. Salvando el hecho de que toda clase que implemente este estándar debe ofrecer un constructor vacío, además del constructor con argumentos, estos objetos deben cumplir las siguientes reglas:

- Implementar la interfaz Serializable. La serialización permite a las aplicaciones y frameworks guardar, almacenar y restaurar el estado del objeto JavaBean independientemente de la plataforma. La serialización permite convertir el objeto en una secuencia de bytes para la transmisión asegurando una reconstrucción de los valores de sus atributos en recepción de manera fiable ya que el componente se carga a través del fichero binario serializado. Esta característica junto con el uso de la base de datos relacional, nos permite asegurar la persistencia de los datos del sistema.
- Otro aspecto básico de codificación de cualquier objeto desarrollado según este estándar, es que debe implementar métodos get y set para cada uno de sus atributos. Dado que, cada atributo Java corresponde a un campo de la base de datos, el nivel de acceso a través de los métodos públicos get y set, es capaz de

ofrecer los datos del sistema mapeados en objetos Java cuyos atributos contendrán los valores de la BD. Así, este nivel proporciona un acceso rápido a los datos del sistema al nivel de lógica simplificando el tratamiento de la información a través de objetos Java.

Este tipo de objetos no implementan ninguna lógica adicional al almacenaje y entrega de los valores definidos.

Para poder realizar operaciones y cargar de contenido a estos DTO se necesita un objeto de acceso a datos. Por ello, se ha definido un DAO por cada objeto de transferencia JavaBean.

### 4.4.2. Objetos de acceso a datos

Para implementar el acceso a los datos del sistema, se ha decidido utilizar el patrón DAO. Los objetos DAO se encargar de obtener la información de la fuente de datos, encapsulándola en el DTO adecuado o en una colección de DTOs. Los DAO ofrecen a través de sus métodos la explotación de los datos a la capa de lógica.

Usaremos la API Java Data Base Connectivity (JDBC) para acceder al sistema gestor de la base de datos utilizando el paquete java.sql.*, para lo cual, se ha incluido en el desarrollo la implementación de la interfaz JDBC proporcionada en el driver com.mysql.jdbc.Driver. Esta API permite la ejecución de sentencias SQL sobre una base de datos relacional utilizando una serie de objetos encargados de funciones distintas:

- Objeto gestor de drivers, DriverManager.
- Objeto de representación de la conexión con la base de datos, Connection.
- Objeto que representa la sentencia SQL a ejecutar sobre la base de datos, PreparedStatement.
- Objeto de representación del conjunto de resultados obtenidos tras la sentencia, ResultSet.

Utilizando estos objetos se han implementado varios objetos de acceso a datos todos con una lógica común. Esta lógica está basada en la incorporación de métodos encargados de la gestión de la conexión con la base de datos y métodos destinados al manejo de los datos.

Se ha desarrollado una clase denominada `SonrieDAO`, que implementa todos los métodos comunes al resto de DAOs, es decir, los métodos de acceso a la base de datos. La herencia nos permite definir a partir de esta clase Java distintas subclases dedicadas al manejo de cada una de las tablas concretas definiendo en ellas el resto de métodos necesarios para el manejo de los datos del sistema.

Las siguientes subclases heredan de `SonrieDAO` las funciones relacionadas con la obtención y liberación de conexiones e implementan los métodos de acceso a los datos, delegando las funciones de acceso a la base de datos en los métodos de SonrieDAO. Todos los objetos DAO pertenecen al paquete bdg.operaciones.

- **bdg.operaciones.JuegoDAO** Esta clase se encarga de ofrecer métodos de gestión y modificación de los datos de los juegos del sistema. Además ofrece métodos de consulta en base a diferentes criterios.
- **bdg.operaciones.NinoDAO** encargada de la gestión de los niños del sistema. Sus métodos involucran todas las acciones relacionadas con los datos de los niños, estas son, consulta de datos de un niño o de todos los niños de un determinado profesional, modificación de datos de un niño concreto, alta de nuevos niños en el sistema o eliminación de los datos de un niño.
- **bdg.operaciones.ProfesionalDAO**. Esta clase se encarga de las operaciones relacionadas con los profesionales del sistema: consulta de profesionales con distintos criterios de consulta, así como, modificación, alta y baja.
- **bdg.operaciones.ConfiguraJuegoDAO**. Esta clase proporciona un método para acceder a las configuraciones de los juegos asociadas a cada niño, esto es, para cada juego y fecha el número de veces que debe ejecutarse y el tiempo máximo por intento. Además ofrece otro método que permite añadir una nueva configuración de un juego determinado.
- **bdg.operaciones.RealizaDAO** encargada de ofrecer el conjunto de entradas de la tabla del sistema referente a la realización de cada intento de cada juego. Para cada niño, el sistema almacena el número de intento y el juego en cuestión, informando acerca de la realización positiva o negativa de un determinado intento, incluyendo el uso de otros músculos para llevarlo a cabo.
- **bdg.operaciones.CentroDAO**. Esta clase implementa métodos de consulta de centros en función del nombre o del identificador del centro.
- **bdg.operaciones.EspecialidadDAO**, encargada de ofrecer métodos de consulta de las especialidades de los profesionales del sistema.

  En la Figura 24 se muestra el diagrama de clases de las clases DAO del sistema indicando esta relación de herencia.

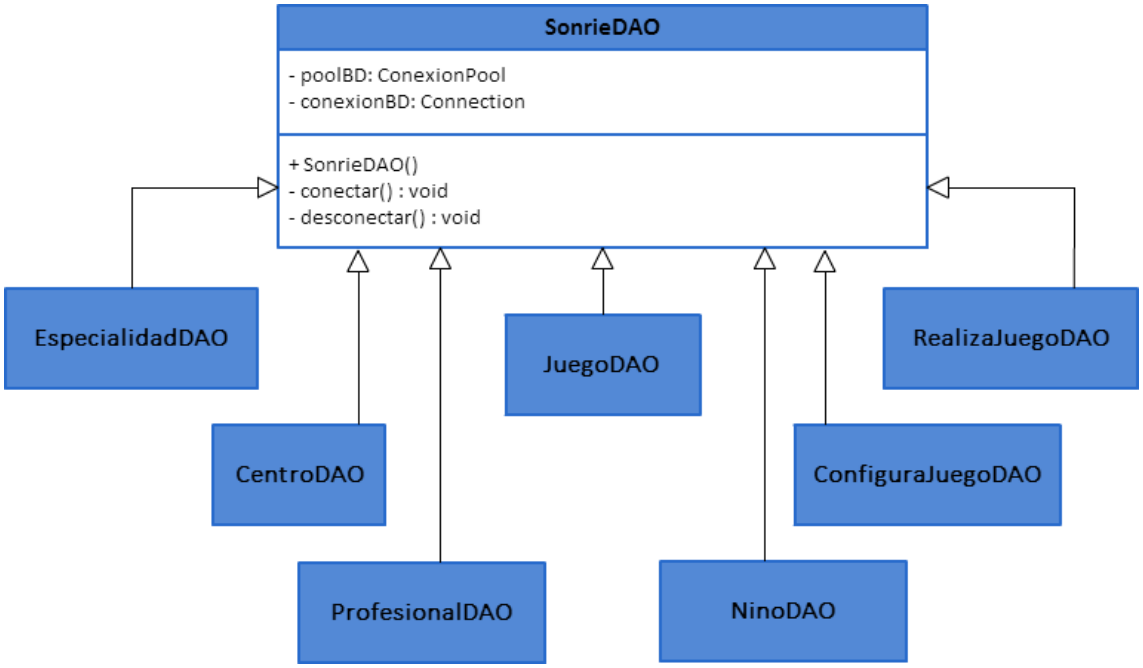


*Figura 24. Jerarquía de clases DAO*

## Gestión de la conexión con la base de datos

Para poder realizar conexiones concurrentes a la base de datos de forma óptima, se ha implementado un pool de conexiones. El pool mantiene un conjunto de conexiones abiertas con la base de datos y las facilita según se van demandando. Cuando los DAOs que actúan como clientes del pool, solicitan una conexión, el pool busca una conexión libre y la adjudica. Una vez realizada la operación, el objeto DAO cierra la conexión. El pool al recibir la petición de cierre no cierra la conexión, si no que la marca como libre para otra posible demanda. Esto es de vital importancia, pues si no se invoca el cierre de la conexión, el pool asume que la conexión sigue ocupada por el cliente que la solicitó.

De esta forma el pool mantiene varias conexiones abiertas, que va sirviendo y marcando como utilizadas, pudiendo reaprovecharlas en llamadas posteriores una vez etiquetadas como libres. En caso de no tener conexiones disponibles en el momento de su invocación, el pool crea una nueva conexión y la devuelve a su cliente, aunque el número de conexiones que pueden asignarse en el mismo momento puede limitarse. Este hecho favorece la concurrencia y la rapidez en el acceso, ya que se podrán utilizar conexiones concurrentes y simultáneas para los distintos clientes y acciones, que, además, ya están establecidas cuando se demandan.

La implementación se basa en el uso de la API JDBC y en la definición de un conjunto de parámetros necesarios para obtener la conexión. En primer lugar, debemos seleccionar el driver adecuado, en este caso, dado que nuestra base de datos es de tipo MySQL, utilizaremos el driver JDBC para MySQL, denominado MySQL Connector/J y disponible en

`com.mysql.jdbc.Driver` a través de la inclusión de la librería mysql-connector-java-5.1.34.bin.jar en nuestro proyecto. La selección del Driver y otros parámetros necesarios para la conexión serán atributos de un objeto Java `DataSource (javax.sql.DataSource)` encargado de implementar el pool de conexiones. Este objeto se construye a través de un objeto de tipo `BasicDataSource (Apache Commons BDCP)`, con los siguientes parámetros:

- `DriverClassName=com.mysql.jdbc.Driver`. Como se ha comentado debemos incluir el driver adecuado para la base de datos en nuestro caso, MySQL.
- URL de la base de datos. Este campo es importantísimo pues el acceso hacia la base de datos se realiza siempre a través de URIs. Nuestra URL es la siguiente:

  `jdbc:mysql://localhost:3306/bd_sonrie`

  Compuesta por los siguientes parámetros:
  - Transporte: JDBC.
  - Tipo de base de datos: MySQL.
  - Servidor: localhost.
  - Puerto: 3306.
  - Base de datos: bd_sonrie.
  - Además para establecer la conexión necesitamos introducir el usuario y la contraseña de la base de datos. Estos parámetros se obtienen de un fichero de configuración que se describirá con mayor detalle en el capítulo dedicado a los archivos de configuración del sistema.

Utilizando estos objetos se ha construido la clase `ConexionPool` encargada de instanciar la conexión cuando se construye y de ofrecer un método `extraerConexion()` que obtiene una conexión del pool devolviendo un objeto de tipo `Connection`, y, un método `liberarConexion()` que devuelve la conexión, que recibe como parámetro, al pool.

La clase `SonrieDAO` construye un objeto `ConexionPool` e implementa métodos para utilizar los métodos de conexión mencionados, además, incluye el tratamiento de las posibles excepciones SQL en caso de no poder crear o liberar conexiones. El resto de objetos de acceso a datos disponen de los métodos heredados de `SonrieDAO` para obtener y devolver una conexión al pool.

Además de la gestión de la conexión, necesitamos que los DAOs sean capaces de manejar los datos de la base de datos. Para ello se han implementado en las clases derivadas, métodos destinados al manejo de datos descritos a continuación.

## Métodos destinados al manejo de datos

Se han desarrollado métodos para el tratamiento de datos que implementen las cuatro acciones básicas SQL:

- **Métodos de consulta de datos**: encapsulan sentencias de tipo SELECT de SQL. Este tipo de métodos pueden estar especializados en consultas de todas las entradas de una tabla o, pueden estar parametrizados según algún criterio de selección. Las sentencias que realizan selecciones por un campo de clave primaria, devolverán el DTO constituido por los valores de los campos del registro identificado por esa clave. Si la selección es por campos que no son clave devolverán una colección de objetos de transferencia de datos que cumplan con el valor del campo introducido.
- **Métodos de inserción de datos**: encapsulan las sentencias INSERT de SQL y reciben como parámetro el objeto de transferencia que contiene los datos del registro a insertar.
- **Métodos de actualización de datos**: De manera análoga, los métodos de modificación o actualización encapsulan sentencias UPDATE de SQL, recibiendo como parámetro el objeto que contiene los nuevos valores de la entrada que se quiere actualizar.
- **Métodos de borrado de datos**: encapsulan sentencias SQL de tipo DELETE y reciben la clave primaria o un conjunto de valores del registro a eliminar.

Todos estos métodos se implementan según una lógica común:

- En primer lugar, utilizamos un objeto de tipo `PreparedStatement` para cargar la sentencia. Este tipo de objetos nos permite precompilar la sentencia y aceptar parámetros de entrada en tiempo de ejecución. Una vez definida la sentencia, se procede con su ejecución a través del método `execute()`.
- En el caso de las sentencias de consulta de datos, utilizamos el método `executeQuery()`. Este método nos devuelve el conjunto de resultados según el criterio de selección introducido en forma de ResultSet. En este punto, el objeto DAO debe construir un bean o una colección de ellos a partir de este ResultSet. El DAO se encarga de introducir el valor adecuado en cada atributo del objeto DTO a través de métodos getString o getInt con parámetro el nombre del campo de la tabla correspondiente. Una vez construido, el DAO retorna este DTO o un conjunto de ellos.
- El resto de operaciones se han implementado de tipo void si se considera que no se necesita retorno alguno o de una forma similar a la anterior si la aplicación necesita recibir el objeto DTO.

De esta forma, la capa de lógica al instanciar un objeto de tipo DAO obtiene una conexión hacia la base de datos y la disponibilidad del uso de los métodos implementados para realizar consultas, inserciones, actualizaciones o borrado de datos de la base de datos. Es importante destacar que esta capa debe encargarse de liberar la conexión una vez ejecutada la sentencia SQL para liberar recursos, y que, una vez recibido el DTO o conjunto de DTOs, puede realizar un acceso rápido a los datos a través de los métodos get del DTO.

En la Figura 25 podemos visualizar la arquitectura de componentes software de la capa de acceso, involucrados en el proceso de mantenimiento de los datos de los niños del sistema según el modelo descrito. Serán necesarias las clases: `SonrieDAO` junto con `ConexionPool` para obtener y liberar la conexión, `NinoDAO` para ejecutar las sentencias y construir el objeto DTO, `NinoBean`, en función de los valores recibidos de la base de datos. Enviando este `NinoBean` o conjunto de ellos al nivel superior ofreciendo un tratamiento de datos muy sencillo. El resto de funcionalidades del sistema se ha implementado de forma análoga.

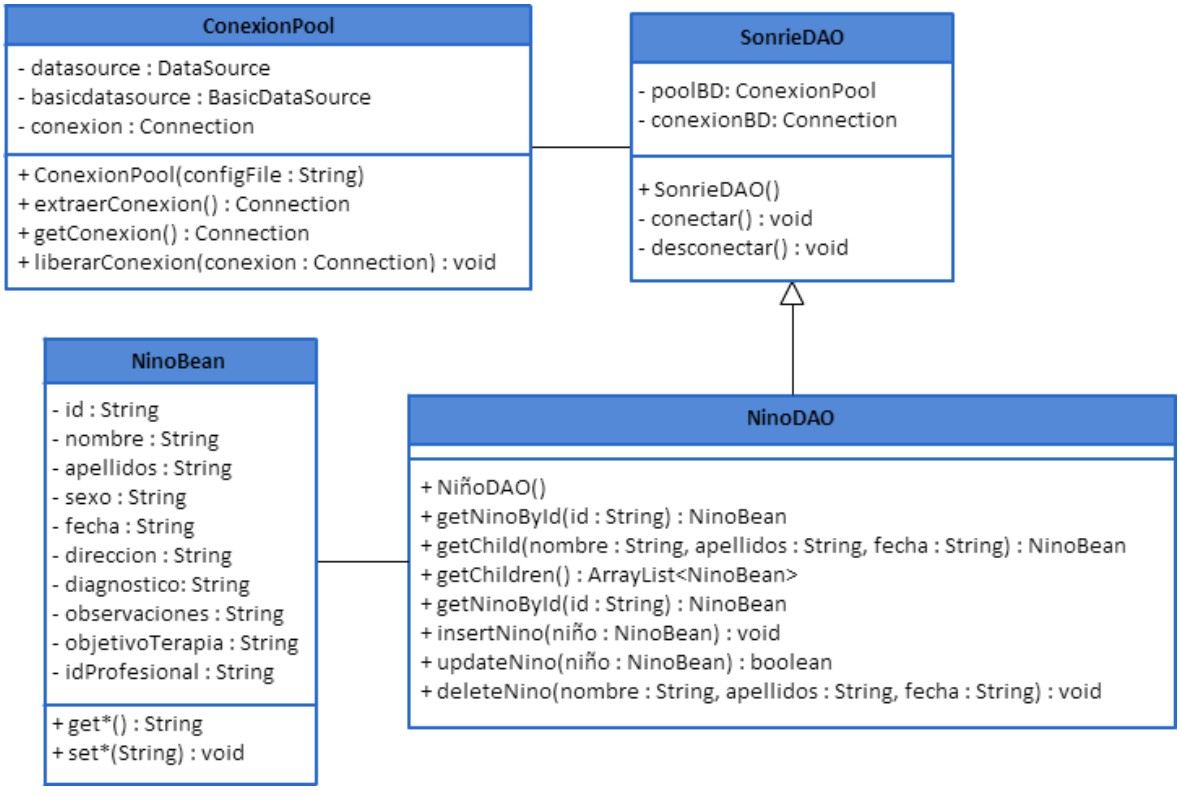


*Figura 25. Clases de la capa de acceso a datos para la gestión de niños*

## 4.5.Diseño y construcción de la lógica de negocio

Esta capa incluye toda la lógica necesaria para ofrecer los servicios requeridos por la aplicación utilizando el nivel de acceso a datos. Para implementar este nivel, se ha decidido utilizar un servicio web. A nivel conceptual, un servicio web es un componente software que proporciona acceso a otros elementos de la red, en nuestro caso a la capa de acceso a los datos. La aplicación cliente consume el servicio web usando mensajes de solicitud y respuesta básicos, lo que conlleva asumir muy pocas suposiciones sobre las capacidades tecnológicas del receptor.

Los servicios web son aplicaciones cliente-servidor que se comunican a través del protocolo de transferencia de hipertexto (HTTP). Según lo descrito por el World Wide Web Consortium (W3C), los servicios web proporcionan un medio estándar de interoperabilidad entre las aplicaciones de software que se ejecutan en una variedad de plataformas y marcos [W3C15]. Los servicios Web se caracterizan por su gran interoperabilidad y extensibilidad, así como por sus descripciones procesables por máquinas, gracias al empleo de XML.

A nivel técnico, existen numerosas implementaciones posibles para construir servicios web en la plataforma Java EE. Los dos tipos más utilizados son los servicios web SOAP y los servicios web RESTful [ORA13].

- **Servicios web SOAP (Simple Object Access Protocol)**

  Este tipo de servicios web utilizan mensajes XML basados en el protocolo de acceso a objetos SOAP. Los tipos de mensajes SOAP intercambiados entre cliente y servidor y sus formatos están también definidos en XML. Las operaciones que ofrece el servicio aparecen descritas en el Web Services Description Language (WSDL), un lenguaje XML para la definición de las interfaces sintácticamente. Este tipo de servicios ofrece dificultades a la hora de escalar pues, si se quiere modificar el servicio o añadir nuevas funcionalidades deberíamos en primer lugar modificar la clase Java, para posteriormente generar el WSDL descriptor del servicio y posteriormente las clases que utilizaremos en el cliente. Con SOAP se permite crear un solo servicio con varios métodos.

- **Servicios web REST (Representational State Transfer)**

  Este tipo de servicios web, conocidos como RESTful siguen el estilo de arquitectura REST. Utilizan directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre ellos en formato XML o JSON sin tener en cuenta las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes petición-respuesta como ocurría con SOAP. Cualquier elemento puede ser identificado como un recurso y cada recurso identificado unívocamente a través de una URI. Por tanto es mucho más sencillo de integrar en HTTP. Las redes sociales Twitter o Facebook están basados en RESTful.

Los servicios web SOAP son quizá más sencillos de implementar, pero mucho más complicados de consumir, pues necesitamos toda una API para construir las clases del cliente utilizando el WSDL. Además, este hecho implica que, si se hacen modificaciones en el servidor, hay que volver a generar las clases del cliente. Con servicios web de tipo REST es mucho más sencillo acceder a los recursos desde el cliente web, pues únicamente hay que recurrir a su URI, sin necesidad de utilizar clases auxiliares en el cliente.

Teniendo en cuenta los requisitos del sistema en cuanto a escalabilidad, y sencillez en el lado del cliente, se ha decidido implementar un web service RESTful que desplegaremos en nuestro contenedor web Tomcat. Para desarrollar el servicio web se ha utilizado la API JAX-RS de Java, diseñada para facilitar el desarrollo de aplicaciones que utilizan la arquitectura REST. Esta API utiliza anotaciones Java para simplificar el desarrollo de servicios web RESTful. Las anotaciones en Java, permiten añadir metadatos al código fuente que se cargarán en tiempo de ejecución. De esta forma, podemos describir, por ejemplo, la ruta de cada método del WS utilizando la anotación @Path. Usando estas anotaciones se definirán los recursos y las acciones que se pueden realizar en esos recursos. Se ha utilizado Jersey, la implementación de referencia de la especificación JAX-RS. Jersey implementa soporte para las anotaciones definidas en la especificación JAX-RS, lo que facilita a los desarrolladores crear servicios web RESTful en Java. Los recursos están expuestos a los clientes mediante la implementación del WAR correspondiente al servicio web en un servidor web, en nuestro caso Apache y su contenedor TOMCAT.

Se necesita de una única clase para implementar el servicio, que ofrecerá una serie de métodos públicos de alguno los siguientes tipos, que tienen una correspondencia directa con los métodos HTTP clásicos:

- GET: Para obtener un valor. Puede ser un listado de objetos
- POST: Para guardar un nuevo objeto en la aplicación
- DELETE: Para eliminar un objeto
- PUT: Para actualizar un objeto

El servicio web implementa un método público por cada método de cada objeto DAO. En cada uno de los métodos serán necesarias una serie de anotaciones para que el nivel de aplicación pueda consumir el servicio web y sus métodos, ya que estas anotaciones permiten mapear las clases recursos en recursos web. Se comentarán las anotaciones utilizadas en este proyecto, pues existe un gran repertorio de ellas:

- @GET, @PUT, @POST, @DELETE y @HEAD especifican el tipo de petición HTTP de un recurso
- @Path especifica la ruta de acceso relativa para una clase recurso o método
- @Produces especifica los tipos de medios MIME de respuesta

- Además, se necesita una anotación adicional para pasar los parámetros a aquellos métodos que lo necesiten. La notación @PathParam enlaza un parámetro a la ruta del método

Para comprender de manera sencilla estas definiciones de los métodos del servicio web implementado, necesarias para que la aplicación pueda consumirlo, partiremos de un ejemplo concreto: el método denominado getTodosProfesionales().

Este método está definido por las siguientes anotaciones:

- @GET
- @Produces("application/json")
- @Path("profesionales")

Esto significa que la aplicación consumirá un método de tipo get que devolverá la información en formato JSON. Para ello deberá invocar a la siguiente URL: http://localhost:8080/Sonrie/ws/profesionales.

La URL está compuesta por los siguientes elementos:

- Dirección y puerto del servidor donde se aloja el servicio, en nuestro caso localhost:8080 ya que se trata del puerto del Tomcat.
- Contexto de la aplicación. En nuestro caso Sonrie. Este valor suele coincidir, y en nuestro caso así lo hacen, con el nombre del proyecto. Y será necesario definirlo en el descriptor de despliegue web.xml que se muestra en el apartado dedicado a ello.
- ws es el nombre o ruta de nuestro recurso, es decir, el WS. Este nombre se ha definido con la anotación @Path(“/ws”) en la definición de la clase pública que implementa el WS.
- Profesionales corresponde al Path definido para consumir el método getTodosProfesionales() de nuestro recurso web.

Con estos parámetros se identifica unívocamente el método de forma que la aplicación web pueda consumirlo. Existen algunos métodos que necesitan recibir parámetros por parte de la aplicación por tanto debemos utilizar la anotación @PathParam descrita. El siguiente ejemplo muestra como:

En caso de algunos métodos, como los de inserción, por ejemplo, insertChild() que se encarga de insertar en la base de datos un nuevo niño, necesitamos conocer los valores de todos y cada uno de los campos correspondientes al bean que se debe construir para

después mapearlo en un registro de la tabla niño. Para ello, nuestro WS queda definido de la siguiente forma:

```
@POST
@Path("insertchildren/{id}/{nombre}/{apellidos}/{sexo}/{fecha}/{direccion
}/{diagnostico}/{observaciones}/{objetivo}/{idprofesional}")

public String insertChild(@PathParam("id")String id,
                          @PathParam("nombre")String nombre,
                          @PathParam("apellidos")String apellidos,
                          @PathParam("sexo")String sexo,
                          @PathParam("fecha")String fecha,
                          @PathParam("direccion")String direccion,
                          @PathParam("diagnostico")String diagnostico,
                          @PathParam("observaciones") String observaciones,
                          @PathParam("objetivo")String objetivo,
                          @PathParam("idprofesional")String idprof){}
```

Para consumir este recurso la aplicación invocará una operación de tipo POST a la siguiente URL: http://localhost:8080/Sonrie/ws/insertProfesional/id/nombre/apellidos/..

Es decir se deben concatenar todos los valores asociados al niño que hayan sido introducidos vía web, separados por barras:

```
@Path("insertchildren/{id}/{nombre}/{apellidos}/{sexo}/{fecha}/{direccion
}/{diagnostico}/{observaciones}/{objetivo}/{idprofesional}")
```

En otro tipo de métodos, los de actualización, necesitamos conocer también todos los valores de los campos de la tabla correspondiente. Por ejemplo, el método updateJuego(), que se encarga de actualizar los valores definidos para un juego concreto, construye un bean con todos los atributos correspondientes a un juego, es por ello, que la definición del método es la siguiente:

```
@POST
@Path("updateJuego/{id}/{nombre}/{objetivo}/{repeticiones}/{tiempo}")

public void updateJuego(@PathParam("id")String id,
                        @PathParam("nombre")String nombre,
                        @PathParam("objetivo")String objetivo,
                        @PathParam("repeticiones")int repeticiones,
                        @PathParam("tiempo")int tiempo){}
```

Para consumir este recurso la aplicación invocará una operación de tipo POST a la siguiente URL: http://localhost:8080/Sonrie/ws/updateJuego/id/nombre/objetivo/repeticiones/tiempo .

Los métodos de eliminación definidos, reciben como parámetros un conjunto de datos que determinan el registro de la tabla que se quiere eliminar. Por ejemplo, el método darBajaNino(), recibe el nombre, apellidos y fecha de nacimiento del niño que se quiere dar de baja del sistema para identificarlo correctamente en la BD. La definición del método es la siguiente

```
@DELETE
@Path("darBajaNino/{nombre}/{apellidos}/{fecha}")
public void darBajaNino(@PathParam("nombre")String nombre,
                        @PathParam("apellidos")String apellidos,
                        @PathParam("fecha")String fecha) {}
```

Por tanto, la aplicación invocará una operación de tipo DELETE a la siguiente URL: http://localhost:8080/Sonrie/ws/darBajaNino/nombre/apellidos/fecha

En cuanto al formato de respuesta que debe ofrecer el servicio web, se ha decidido utilizar el formato JSON. Esta conversión se realiza de manera muy sencilla, mediante la librería de Java Gson proporcionada por Google. Esta librería se utiliza para convertir objetos serializados Java en su representación en formato JSON y viceversa [WIK15]. En nuestro sistema esta librería se usa en el servicio web para transmitir la información recibida a través del DAO en un formato adecuado a la capa de presentación del sistema, simplemente utilizando dos instrucciones:

```
Gson gson = new Gson();
String json = gson.toJson(resultado);
```

Donde la variable resultado contiene el bean o el conjunto de DTOs devueltos tras la sentencia SQL. Se ha elegido el formato JSON, subconjunto de JavaScript, por su sencillez en contraposición a XML.

JSON es un formato ligero de intercambio de datos independiente del lenguaje, y, en general, menos pesado que XML [ECM13]. Es un formato compuesto por texto plano, por tanto ofrece facilidad de lectura y escritura en una sintaxis muy simple. JSON posee una estructura jerárquica, lo que permite el intercambio de datos complejos. Este formato está constituido por dos tipos de estructuras, una colección de pares nombre-valor, que se tratará como un objeto (Figura 26) y una lista ordenada de valores o de objetos que se tratará como un array (Figura 27).

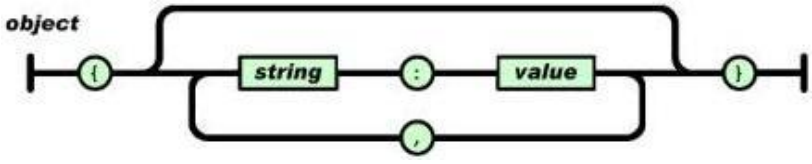


*Figura 26. Object JSON*

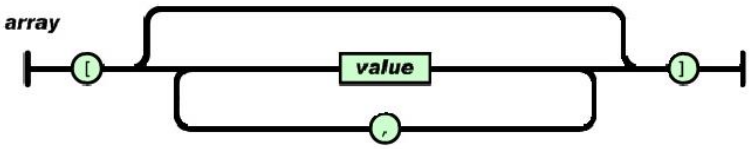


*Figura 27. Array JSON*

El siguiente ejemplo corresponde a una barra de menús definida en formato JSON, donde podemos ver el objeto menú, compuesto por varios pares atributo-valor. Uno de estos valores, menuitem, corresponde a un array de valores [WIK15].

```
{"menu": {
   "id": "file",
   "value": "File",
   "popup": {
     "menuitem": [
       {"value": "New", "onclick": "CreateNewDoc()"},
       {"value": "Open", "onclick": "OpenDoc()"},
       {"value": "Close", "onclick": "CloseDoc()"}
     ]
   }
}
}
```

Como vemos JSON es un formato sencillo de descripción e intercambio de datos y existen herramientas rápidas como Gson que actúan como parsers. Lo más interesante de este formato es el rápido acceso que proporciona a cualquiera de los valores de la estructura. Esto significa, simplificar enormemente el tratamiento de la información en el lado del cliente una vez recibido el documento JSON. Invocando a la URI adecuada, por ejemplo, la petición de consulta de todos los niños del sistema, se recibe un documento que contiene un conjunto de objetos definidos de forma equivalente al objeto anterior. Los pares atributo-valor corresponderán a los atributos y valores de estos de cada niño, de forma que nuestro documento será un array de NinoBeans, en formato JSON, compuestos por pares atributo valor. La lectura de estos datos al utilizar JavaBeans y JSON combinados es sencillísima, únicamente debemos tratar el documento como un array de objetos. Para acceder al valor de un atributo de un objeto concreto se utilizará la notación .atributo de forma equivalente a C, es decir, por ejemplo si recibimos el documento data, y queremos acceder al nombre del primer elemento, la sentencia sería tan sencilla como data[1].nombre, debido a que hemos definido el atributo nombre en la clase Bean y el documento JSON se crea en base a estas definiciones.

## 4.6.Diseño y construcción de la capa de presentación

En la actualidad, el desarrollo de nuevas tecnologías web permite la construcción de interfaces web mucho más usables y vistosas. En la implementación de nuestra aplicación web se ha utilizado el lenguaje por excelencia en la elaboración de páginas web, HTML *HyperText Markup Language* (lenguaje de marcas de hipertexto) [W3C15]. La última versión de este estándar incluye nuevas funcionalidades y facilidades. Los nuevos dispositivos se adaptan perfectamente al lenguaje HTML5. Incluye nuevas etiquetas, atributos y APIs, centrados en la implementación de aplicaciones web, y sobre todo, pretende facilitar el camino hacia la web semántica ofreciendo un etiquetado más completo y una organización del contenido más precisa que las versiones anteriores [GOO15].

Además, la capa de presentación debe encargarse de consumir los recursos de una manera adecuada. Para ello, debemos utilizar la técnica AJAX (JavaScript asíncrono y XML) de JavaScript. AJAX es una técnica de desarrollo que carga la información de forma asíncrona en la página HTML. Añade a JavaScript funcionalidades en el lado del cliente relacionadas con la comunicación con el servidor. Para conseguir está comunicación se ha utilizado JQuery, un framework de desarrollo web Ajax.

JQuery es un framework que encapsula funciones para generar efectos visuales y, lo que es más interesante, para cargar contenido remoto. Las funciones JavaScript que encapsula nos permiten acceder y modificar cualquier elemento del documento web, así como, modificar el aspecto de una página a través de las hojas de estilo (CSS). Además del aspecto, JQuery permite modificar el contenido de la página de manera dinámica, de esta manera, en una única página HTML se pueden implementarse todas las funcionalidades y capas html asociadas, de forma que cuando son demandadas se muestra el contenido en su correspondiente contenedor y se oculta el resto. Esta tecnología nos permite simplificar el desarrollo del sistema que, con tecnologías más antiguas como JSP, obligaría a utilizar una clase JSP por página necesaria en el lado del cliente, en cambio, JQuery nos permite incluir todo el contenido en el mismo documento HTML. La incorporación de funciones para interactuar con el servidor nos permite traer los recursos de una manera muy simple mostrándolos en alguna de las capas implementadas también de forma sencilla.

Este framework incorpora además, una enorme cantidad de widgets muy interesantes que podemos introducir en nuestra página, a través de la biblioteca JQuery IU. Algunos ejemplos de widgets que nos ofrece son diálogos, calendarios, barras de progreso, pestañas… (Ver Figura 28) En el sistema SONRIE se ha utilizado la versión 1.11.3 de JQuery IU.

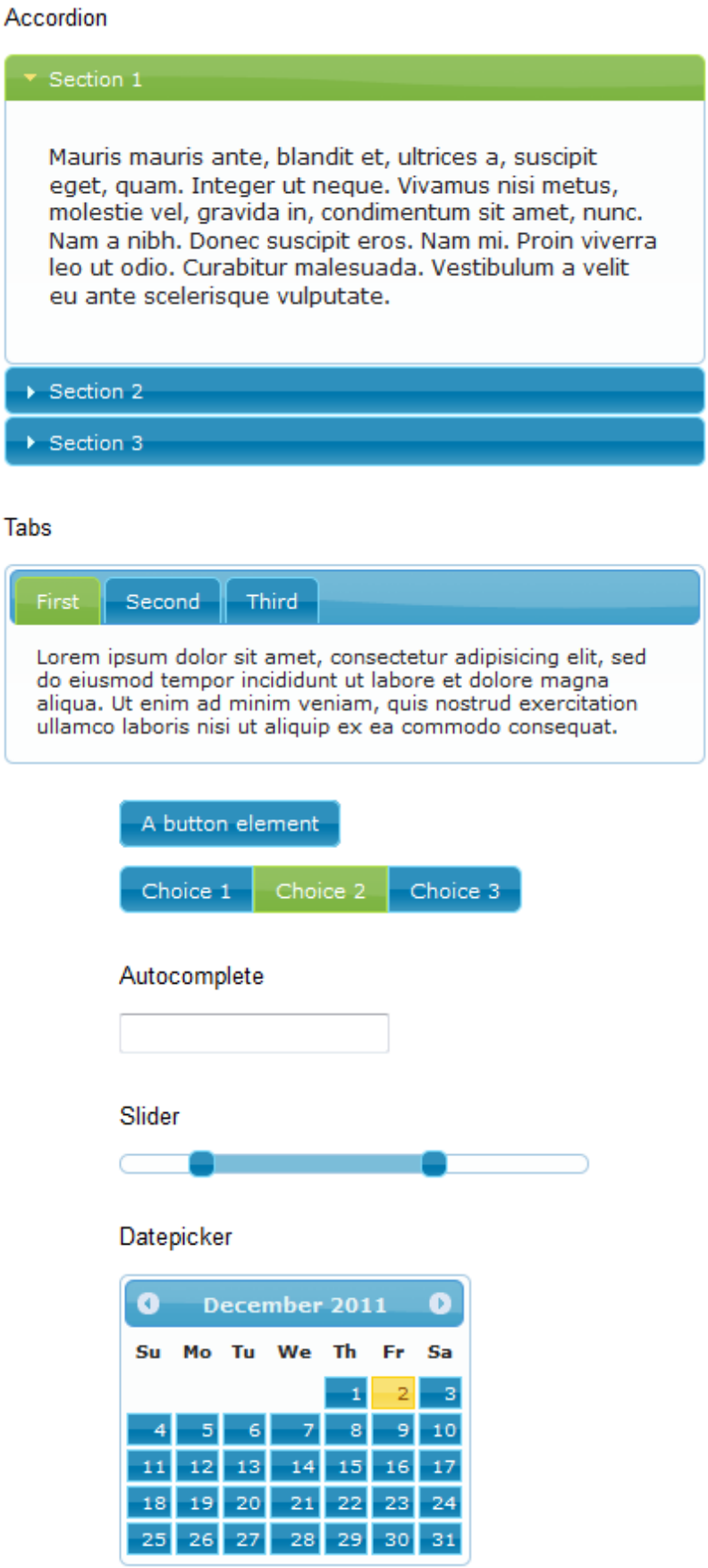


*Figura 28. Widgets JQuery*

No necesitamos programar estos componentes de la interfaz de usuario ya que son de código totalmente libre. Así que, podremos insertarlos fácilmente en nuestro código y utilizar sus funciones JavaScript. Además, podremos modificar los estilos de los widgets de manera sencilla en nuestra hoja de estilos. Lo más interesante que ofrecen estos componentes, aparte de ofrecernos este aspecto tan llamativo y vistoso y ahorrarnos la codificación de todos estos componentes desde cero, es que JQuery nos proporciona muchas funciones para responder a eventos, como por ejemplo, un clic en un botón o la selección de una opción de un menú desplegable. Este hecho ha permitido construir páginas HTML que se van modificando a sí mismas en función de los eventos que el usuario genera en estos widgets.

Se han implementado dos páginas HTML. La primera, sonrie.html, está destinada a los usuarios médicos y ofrece un acceso a la segunda página desarrollada, sonrieadmin.html, diseñada para ser utilizada por el administrador del sistema. Ambas páginas comparten una hoja de estilos común denominada sonrie.css.

Dado que HTTP es un protocolo sin estado, necesitamos algún mecanismo para almacenar ciertos parámetros globales de la aplicación web como los datos de sesión del usuario, en nuestro caso, el identificador del profesional autenticado actualmente en el sistema. Para ello, se ha empleado la propiedad `sessionStorage` que permite acceder a un objeto `Storage` asociado a la sesión actual del navegador del usuario. Esta información almacenada es eliminada al finalizar la sesión de la página, es decir, se mantendrá en las futuras recargas de la página, pero se eliminará al abrir una pestaña o ventana nueva. Antes de HTML5, los datos de las aplicaciones que necesitaban conocerse en todo momento se almacenaban en cookies, y se incluían en cada petición al servidor. SessionStorage nos permite un almacenamiento local en el lado del cliente incluso de grandes cantidades datos sin afectar al rendimiento del sitio web. Además este mecanismo es mucho más seguro que el de las cookies, pues la información no se transfiere al servidor en ningún momento.

Una vez autenticado, la apariencia de las páginas desarrolladas cambia, habilitando los botones que ofrece, siempre comprobando si el objeto sessionStorage contiene un identificador válido en caso del usuario médico o una confirmación positiva en la autenticación en el caso del administrador, pues este, no dispone de un identificador numérico. Para cambiar la apariencia de la página recurrimos a la función css que nos ofrece JQuery. Con ella podemos modificar los estilos de los elementos de la página y por tanto, su visibilidad. Esta función recibe el identificador del componente que se desea visualizar u ocultar. Por ejemplo, en el caso que nos ocupa, una vez autenticado, seguiríamos la siguiente ejecución de acciones:

`$("#autenticacion").css("display", "none");` Para ocultar el menú de autenticación.

`$("#cerrarS").css("display","block");` Para mostrar un botón de cerrar sesión que hasta entonces se mostraba oculto.

`$("#botones").css("display","block");` Para habilitar los botones de la aplicación principal que hasta este momento no desencadenaban ninguna acción en caso de hacer clic sobre ellos.

Este proceso tan sencillo, se utiliza constantemente para modificar el aspecto de la página en función de las demandas del usuario. Por tanto, al contrario de lo que sucedía en JSP que obligaba a implementar tantas vistas como pantallas ofreciese el sitio web, utilizando HTML5 combinado con JQuery conseguimos concentrar toda la información únicamente en una página HTML que se modifica de manera dinámica en función del usuario. Cabe destacar que, la página no puede ser manipulada hasta que no se haya cargado totalmente y por tanto, no puede ejecutarse el código JQuery hasta que el documento esté totalmente cargado y listo. JQuery es capaz de detectar el estado del documento, por ello, es necesario incluir todo el código comentado dentro de la siguiente función de JQuery: `$(document).ready(function(){})`, que solo se ejecuta una vez que el Document Object Model (DOM) está listo. De esta forma, aseguramos que no se intente ocultar un elemento sin haberse cargado previamente

Una vez resuelto el aspecto estético con JQuery, se describirá la mecánica utilizada en las páginas HTML para comunicarse con el servidor. Para ello, se recurrirá a AJAX, una técnica que funciona de forma asíncrona, lo que asegura que los datos adicionales que se solicitan al servidor se cargan en segundo plano sin intervenir en la visualización ni el comportamiento de la página. El uso de esta técnica permite que las páginas se actualicen de forma asíncrona mediante el intercambio de pequeñas cantidades de datos con el servidor en segundo plano, de manera que es posible actualizar una parte de la página con la información según se recibe, sin necesidad de recargar toda la página. En cambio, en las páginas web clásicas, que no utilizan AJAX, se necesita recargar la página entera si se quiere modificar el contenido. Además, AJAX es una técnica válida para múltiples plataformas, sistemas operativos y navegadores ya que está basada en estándares abiertos como JavaScript y DOM [LIB15]. JQuery proporciona varios métodos que nos permiten utilizar la técnica AJAX para solicitar datos a un servidor remoto a través de HTTP, permitiendo además, cargar los datos recibidos directamente en elementos HTML. Para ello necesitamos en primer lugar, identificar cada elemento asociando un valor al atributo **id** del elemento HTML. Utilizaremos la función AJAX de JQuery para realizar peticiones al servidor. Veamos, por ejemplo, el mecanismo utilizado para solicitar los niños a los que atiende un profesional determinado y mostrarlos en una tabla:

```
$.ajax({
            type: "GET",
            dataType: "json",
            url: "http://localhost:8080/Sonrie/ws/children/"+id,
            success: function (data) {
                  var tabla = $("#TablaDatos").dataTable();
                  tabla.fnClearTable();
                              for(var i=0;i<data.length;i++){
                              tabla.fnAddData([
                                          data[i].nombre,
                                          data[i].apellidos,
                                          data[i].fecha,
                              ]);
                   }},
          error: function (error) {
                  console.log(error.responseText);
          });
```

La función AJAX incorpora los atributos `type`, `dataType` y `url` para determinar unívocamente el recurso que se está solicitando. Estos valores corresponden a los valores definidos en el WS en las siguientes anotaciones:

```
@GET
@Produces("application/json")
@Path("children/{idp}")
```

Por tanto, en AJAX el atributo `type` que corresponde al tipo de petición, será GET; el atributo `dataType`, que corresponde al tipo de datos de respuesta del servidor, será JSON y el path corresponderá a la ruta del servidor añadiendo la ruta de este método que resulta ser `children/{idp}` donde, `idp` es un parámetro que corresponde al identificador del profesional autenticado en el sistema y que, como se ha comentado, se obtiene a través del valor almacenado en el objeto `sessionStorage` que se inicializa en el inicio de sesión. Entonces obtenemos la dirección que se incluye en el atributo `url` de la función AJAX `http://localhost:8080/Sonrie/ws/children/"+id`

Al tratarse $.ajax de una llamada asíncrona, esperamos a que reciba la información en el parámetro `data`, para procesarlo y asociar sus valores a la tabla correspondiente. Trataremos el parámetro recibido como un array, pues recordemos que corresponde a un documento JSON que contiene un array de objetos JSON. Entonces, utilizamos las funciones de JQuery para asociar cada valor a una columna de la tabla correspondiente. El acceso a la tabla se realiza a través de su identificador y asociamos a cada columna el valor adecuado de manera sencilla utilizando la sentencia `.nombredelAtributo`, ya que se trata de una definición JSON en la que se mantienen los nombres de los atributos de los objetos DTO para definir los atributos del objeto JSON. Con jQuery es muy sencillo asociar un valor a cada columna, simplemente utilizando la función `fnAddData` introducimos los valores que nos devuelve el web service según el orden en el que hemos declarado las columnas de la tabla.

Para las funciones de inserción o modificación el mecanismo es el mismo, aunque será necesario recoger los valores que el usuario ha introducido y concatenarlos en el atributo `url`. Para recoger estos valores se utiliza el identificador del elemento y la función `.val()` que devuelve el texto que contiene dicho elemento. Una vez obtenidos los valores construimos la URL adecuada concatenando los valores en el orden correcto separados por barras.

## 4.7.Gestión de errores

Dado se ha trabajado en la plataforma Java EE, la gestión de errores, en última instancia, consiste en la gestión de excepciones del sistema. En la implementación del sistema se ha definido una única excepción denominada **`SonrieException`**, que recibe en su constructor un mensaje informando sobre el error o la situación distinta al procesado normal que se haya producido en la aplicación.

Cuando se produce una excepción es capturada y se crea y se lanza una `SonrieException`, que almacena el error producido. Esta excepción se propaga hasta el servicio web.

Las excepciones que pueden producirse son de tipo `SQLException`. El servidor web analiza el tipo de fallo e informa al usuario en caso necesario con un mensaje que se muestra en la aplicación web. Esto ocurriría, por ejemplo, en caso de introducir alguna clave primaria repetida. En este caso el SGBD no aceptaría el valor devolviendo una `SQLException` . Entonces la aplicación web solicitaría la introducción de un nuevo valor para ese campo.

El WS analiza el Bean recibido. Si se produce la excepción se devuelve el objeto NinoBean instanciado con el constructor vacío anteriormente a la ejecución de la sentencia. El WS analiza el identificador. En caso de ser un espacio en blanco, lo que correspondería al valor asignado con el constructor vacío, asigna el valor “repetido” al id del NinoBean que recibirá la aplicación. Esta se encarga de analizarlo para en caso de haber intentado insertar una clave repetida, notificarlo al usuario para que introduzca otro valor en su lugar con la siguiente secuencia de acciones:

```
url: "http://localhost:8080/Sonrie/ws/insertchild/"+id+"/"+n+"/"+a+"/"
+s+"/"+fecha+"/"+dir+"/"+diag+"/"+obs+"/"+obj+"/"+idprof,
success: function (data) {
            var id = data.id;
            if(id == "duplicado") {
            $("#resumen").text("ERROR: El identificador introducido ya
            existe en el sistema. Por favor elija otro valor.");
            $("#añadir").css("display", "block");
            $("#consultar").css("display", "none");
```

De esta forma, se muestra el mensaje de error al usuario junto con los elementos encargados de recoger los datos a insertar en la BD.

## 4.8. Archivos de configuración

Los archivos de configuración del sistema permiten la modificación de ciertos parámetros que afectan al sistema en su conjunto sin necesidad de modificar el código de la aplicación.

Existen diferentes archivos de configuración en el sistema, con utilidades diferenciadas, estos son:

En primer lugar, el archivo **conexionBD.properties**, cuya misión es clave para el funcionamiento del sistema. En él se almacenan los datos necesarios para poder acceder a la base de daos del sistema. Estos datos son la URL, el usuario y la password de la base de datos. De manera que, cuando se crea el pool de conexiones se accede a este archivo para obtener los datos de conexión hacia la base de datos.

El segundo de los archivos de configuración necesarios, es el archivo **web.xml.** Es el descriptor de despliegue del sistema y más concretamente del servicio web. Contiene toda la información necesaria para que el contenedor web pueda desplegar el servicio. Jersey, la implementación de WS REST utilizada, proporciona un servlet que escanea las clases predefinidas para identificar los recursos REST. Por tanto en el archivo web.xml necesitamos definir este servlet y la localización de la clase que implementa el WS REST. En este caso, `<param-value>` contiene la ruta del WS que corresponde al paquete en el que está almacenado. A continuación se muestra el contenido del archivo web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"
version="3.1">
  <display-name>Sonrie</display-name>
  <servlet>
    <description></description>
    <display-name>RestServlet</display-name>
    <servlet-name>RestServlet</servlet-name>
    <servlet-
class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-
class>
    <init-param>
      <param-name>com.sun.jersey.config.property.packages</param-name>
      <param-value>bdg.webservice</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>RestServlet</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Como vemos en el archivo de configuración también se define en el campo <display-name>, que almacena el contexto de nuestro servicio, o lo que es lo mismo, la ruta que deberá invocarse para solicitar alguna petición. Este será el primer valor necesario si se quiere invocar al servidor web después de la dirección y puerto de la máquina que aloja el WS desarrollado. Después, tal como se ha comentado será necesario concatenar la ruta del método en cuestión y los parámetros, si es existen, es decir, se utilizará una URL del tipo:

http://localhost:8080/Sonrie/ws/{rutaMétodo}/{valor atributo 1}/…/{valor atributo N}

Existe un último fichero en el que se almacena el hash del usuario administrador. Se ha decidido utilizar este mecanismo en lugar de almacenarlo en la base de datos para posibilitar el cifrado de esta información si se desea.

# 5. Resultados

Una vez descrito el sistema implementado, se procederá a la muestra de los resultados obtenidos en las pruebas de dicho sistema. La navegación completa por la aplicación web queda recogida en el Anexo B.

Algunas pruebas realizadas se han basado en aspectos funcionales del sistema. Con ellas se pretende validar el sistema y demostrar que se han cumplido los objetivos señalados. Se ha verificado, además, en otra batería de pruebas, que se cumplen ciertos aspectos de diseño, relacionados con la usabilidad del framework. Dado que el cliente manejará únicamente la página web del framework, la validación del sistema se ha centrado en la validación de la página web, sin embargo, las pruebas de funcionamiento, engloban al sistema completo, por tanto, implícitamente con las pruebas de funcionalidades se comprueba el correcto funcionamiento de todos los componentes del framework desarrollado.

## 5.1.Pruebas relacionadas con las funcionalidades del sistema

En este apartado se pretende demostrar el correcto cumplimiento de las especificaciones del sistema, refiriéndonos concretamente a las funcionalidades y objetivos descritos en este PFG. A continuación se describirán las pruebas más relevantes a las que se ha sometido el sistema desde un punto de vista funcional. Para cada funcionalidad del sistema se incluirán una serie de capturas, que pretenden demostrar el correcto funcionamiento de cada una de ellas, asegurando finalmente la validación del sistema completo.

### 5.1.1. Autenticación

El primer requisito que se debe asegurar es la autorización explícita en la entrada al sistema. Para ello se han inhabilitado todos los botones disponibles en la página inicial hasta que se autentique un usuario del sistema. Por ello, si los campos del menú de autenticación están vacíos o se introduce una contraseña y/o usuario erróneos, el sistema debe informar al usuario manteniendo el menú de autenticación de acceso al sistema. Con las siguientes figuras se pretende demostrar la consecución de este objetivo en el caso de los usuarios médicos (Figura 29) y en el caso del administrador (Figura 30). En primer lugar se muestra el mensaje que reciben los usuarios si no introducen ningún dato y posteriormente los mensajes asociados a la introducción de un usuario y/o contraseña no válidos. Además de mostrar un mensaje informativo, en ambas páginas, se restringe el acceso a las funcionalidades del sistema hasta que se reciben unas credenciales de autenticación válidas.

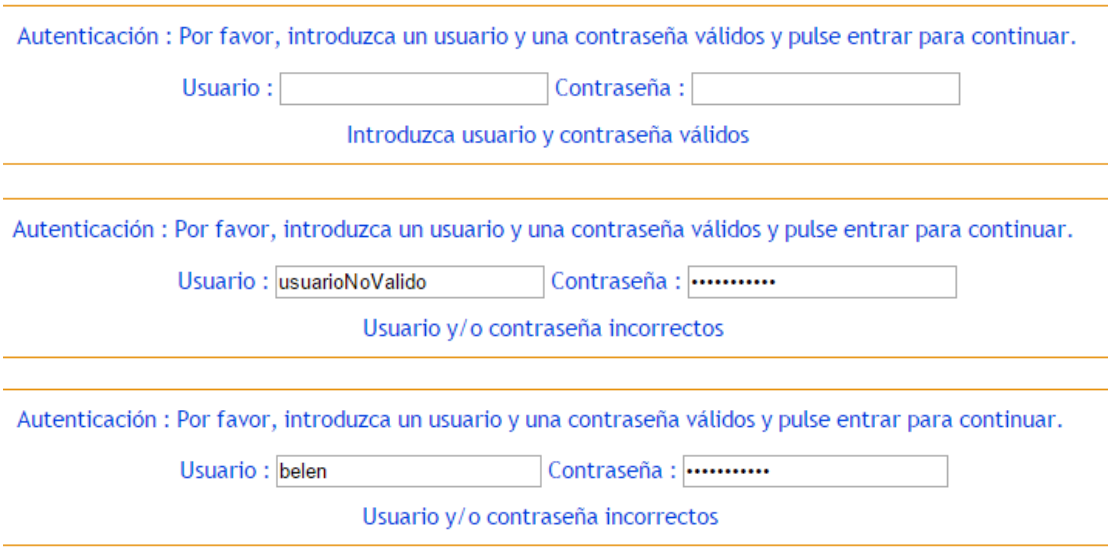


*Figura 29. Errores en el inicio de sesión usuarios médicos*

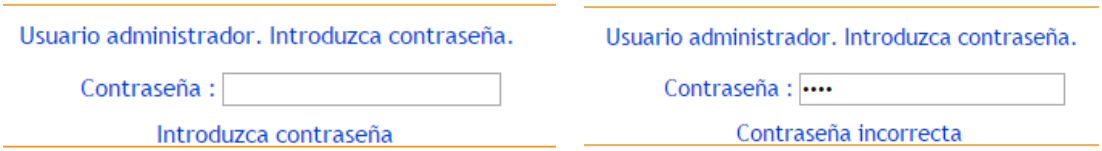


*Figura 30. Errores en la autenticación usuario administrador*

Los botones están deshabilitados para asegurar que ninguna persona realice modificaciones en el sistema, a menos que se trate de un usuario de SONRIE. Igualmente necesario es el cierre de sesión, se ha comprobado que si se pulsa el botón Cerrar sesión disponible en la esquina superior derecha, el sistema regresa a la página inicial en la que los botones se deshabilitan y se muestra de nuevo el menú de autenticación.

Una vez validada la autenticación, se analizarán cada una de las funcionalidades del sistema que recordemos que se resumen en consulta de resultados, modificación de datos y alta y baja de registros.

### 5.1.2. Consulta de resultados

En cuanto a los menús de consulta, cabe destacar que no se muestran todos los datos almacenados en el sistema, sino los más relevantes para identificar unívocamente a cada registro. Se mostrará el menú de consulta de la gestión de niños, sin embargo, estos resultados son extrapolables al resto de opciones de consulta, en los que se muestran tablas similares a la Figura 31. Se ha creído conveniente mostrar este menú de consulta y no cualquier otro, puesto que incorpora una lógica adicional que pretende validarse también en este apartado. Como se ha comentado, cada profesional, una vez autenticado en el sistema, puede visualizar los datos asociados a los niños a los que proporciona tratamiento. Por tanto, Además del correcto funcionamiento de la funcionalidad de

consulta, con esta prueba se pretende demostrar la seguridad del sistema en la gestión de los datos de los niños, que solo se ofrecen al profesional en caso de que sea el profesional encargado de su tratamiento. Por tanto, cada profesional únicamente puede consultar los datos de sus niños, no obteniendo datos acerca del resto de niños del sistema. Para demostrarlo, se han capturado las tablas de consulta desde dos páginas iniciadas con usuarios médicos distintos (Figuras 31 y 32).

Listado de niños

Mostrando 10 resultados

| Nombre | Apellidos | Fecha de nacimiento | Edad |
|---|---|---|---|
| Alejandra | Jiménez Castillo | 27-12-2008 | 6 |
| Belén | González González | 02-02-2007 | 8 |
| Esther | Redondo Alcázar | 09-01-2009 | 6 |
| Javier | Amor Bueno | 18-06-2008 | 7 |
| Javier | García García | 16-07-2009 | 6 |
| Laura | Martínez Gil | 10-10-2010 | 4 |
| Manuel | Pérez Martín | 01-12-2007 | 7 |
| Marta | Martín Bermúdez | 07-12-2007 | 7 |
| Miriam | Muñoz Díaz | 25-5-2005 | 10 |
| Pedro | Jiménez Alcolea | 02-02-2010 | 5 |

Mostrando 1 a 10 de 12 resultados Anterior 1 2 Siguiente

*Figura 31. Consulta de niños profesional 1*

Listado de niños

Mostrando 10 resultados

| Nombre | Apellidos | Fecha de nacimiento | Edad |
|---|---|---|---|
| Alonso | Jiménez Martín | 25-5-2010 | 5 |
| Ana | Garrido Romero | 26-05-2006 | 9 |
| Andrea | Sánchez Ruiz | 06-01-2009 | 6 |

Mostrando 1 a 3 de 3 resultados Anterior 1 Siguiente

*Figura 32.Consulta de niños profesional 2*

Cabe destacar que estas tablas, así como, el resto de tablas del sistema, pueden ordenarse bajo cualquier criterio de los recogidos en cabecera de la tabla, además, se ha implementado un mecanismo de paginación, para ofrecer una mayor comodidad a los usuarios en la consulta de los datos del sistema. Estos aspectos se consideran relacionados con la usabilidad del framework, por tanto, las pruebas realizadas se recogen en el apartado dedicado a tal fin en este mismo capítulo.

### 5.1.3. Inserción de nuevos datos en el sistema

Otra funcionalidad clave es la inserción correcta de datos en el sistema, por ello se muestran las siguientes capturas resultado del alta de un nuevo niño en el sistema, aunque el proceso es similar, y se ha validado igualmente, en el resto de menús de gestión que como se ha descrito corresponden a la gestión de profesionales, de juegos y de terapias. En la Figura 33 se muestran los datos que se pretenden insertar en la tabla niño de la BD y, además, se pretende visualizar una facilidad añadida en este menú. Para evitar errores en la introducción de los datos, se ofrecen unas listas desplegables en las que se podrá seleccionar el centro y la especialidad del nuevo profesional, entre un conjunto de valores que indican las especialidades y centros registrados en el sistema. De esta manera, además de ofrecer una mayor comodidad al usuario en la selección de estos campos, aseguramos que los valores introducidos sean válidos, evitando así posibles errores derivados de la escritura incorrecta de estos campos, hecho que supondría que el nuevo registro no se podría insertar en el sistema.

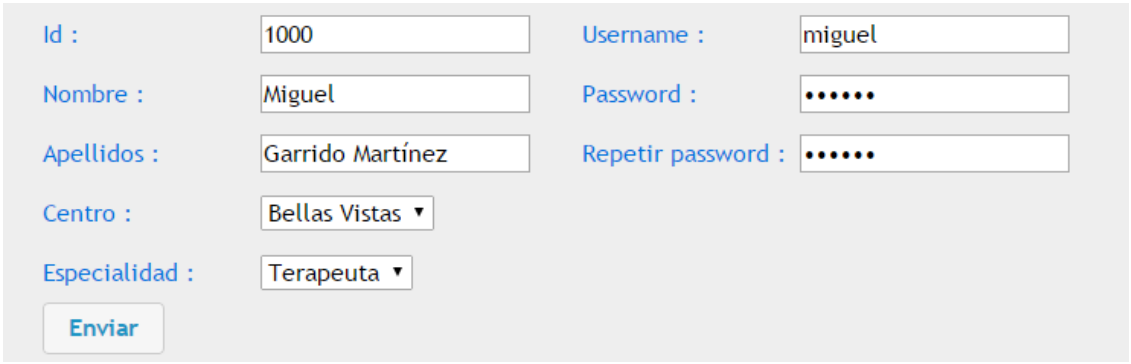


*Figura 33. Alta de nuevo niño en el sistema*

Una vez pulsado el botón enviar, el sistema mostrará la tabla de consulta de profesionales del sistema, si el identificador introducido no existe en el sistema. En caso contrario se informará del error, ya que, obviamente, no se admiten claves primarias repetidas. Si la inserción es válida, se ha comprobado que la tabla de consulta que aparece inmediatamente después de pulsar enviar, muestra el nuevo profesional dado de alta en el sistema (Figura 34).

| Nombre | Apellidos | Centro | Especialidad |
|---|---|---|---|
| Miguel | Garrido Martínez | Bellas Vistas | Terapeuta |
| Rocío | Mir Méndez | Bellas Vistas | Terapeuta |
| Rodrigo | Agudo Fernández | Bellas Vistas | Terapeuta |

Mostrando 11 a 13 de 13 resultados Anterior 1 2 Siguiente

*Figura 34. Prueba de alta de profesional en el sistema*

### 5.1.4. Modificación de datos del sistema

Continuando con las pruebas de las funcionalidades del sistema, se pretende demostrar el correcto funcionamiento de la funcionalidad de modificación de datos. Para ello se ha recurrido a la gestión de niños, pero como en el resto de apartados, los resultados son extrapolables y se han comprobado en el resto de menús de gestión.

Una vez seleccionado el registro que queremos modificar pulsamos el botón Modificar (Figura 35). Entonces, se mostrará en la misma pantalla, un formulario en su lugar (Figura 36) donde el profesional puede visualizar los valores de los campos actualmente, y modificarlos a su antojo.

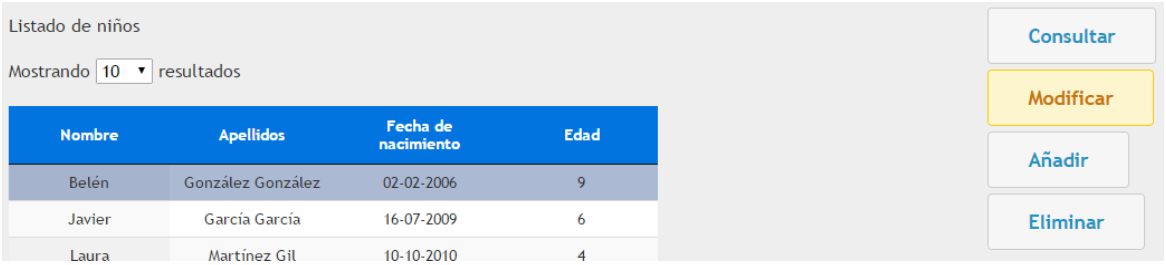


*Figura 35. Proceso para modificar datos*

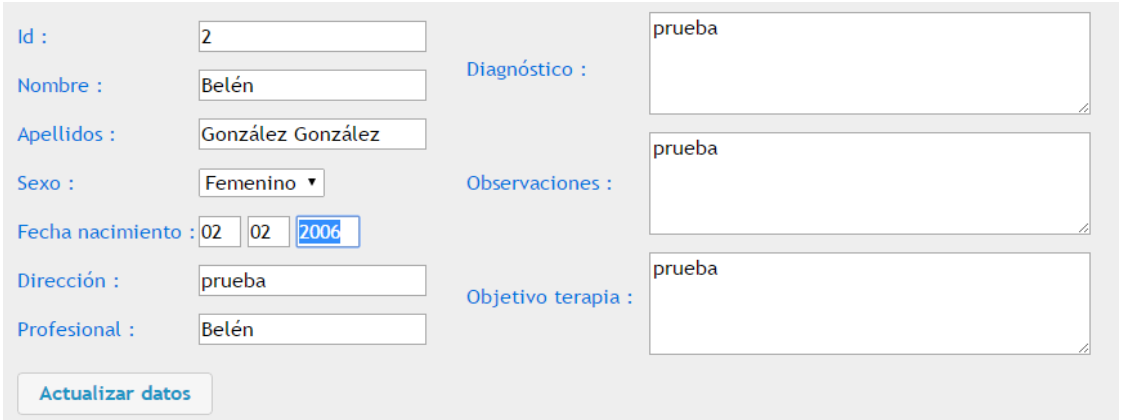


*Figura 36. Modificación de datos*

En esta prueba, se ha modificado el año de nacimiento del niño, para así mostrar también, que el cálculo de la edad del niño que se realiza, se obtiene correctamente (Figura 37).

| Nombre | Apellidos | Fecha de nacimiento | Edad |
|---|---|---|---|
| Belén | González González | 02-02-2007 | 8 |
| Javier | García García | 16-07-2009 | 6 |

*Figura 37. Prueba de modificación de datos*

### 5.1.5. Dar de baja datos del sistema

La última funcionalidad a validar es la eliminación de un registro de una tabla, se ha elegido la tabla profesionales, pero, esta funcionalidad está disponible y validada en el resto de menús de gestión del sistema. Una vez seleccionado el profesional que se quiere dar de baja, pulsamos el botón eliminar (Figura 38)


*Figura 38. Proceso para eliminar datos*

Inmediatamente después la tabla anterior se actualiza. Se ha comprobado que el profesional seleccionado ha sido dado de baja del sistema (Figura 39) y por tanto de la BD.

Con este resultado quedan validadas todas las funcionalidades del sistema, pues todas pueden englobarse en alguno de los apartados descritos en este apartado. A continuación de muestran las pruebas relacionadas con aspectos de diseño y usabilidad del sistema.

| Nombre | Apellidos | Centro | Especialidad |
|---|---|---|---|
| Alberto | Cortázar Castro | Bellas Vistas | Terapeuta |
| Belén | Duro González | Bellas Vistas | Terapeuta |
| Carlos | González Grande | Bellas Vistas | Terapeuta |
| Elena | Rodríguez Sanz | Bellas Vistas | Terapeuta |
| Francisco | Gutiérrez Moreno | Bellas Vistas | Terapeuta |
| Inma | Abella Villa | Bellas Vistas | Terapeuta |
| Laura | Sánchez García | Bellas Vistas | Terapeuta |
| María | Ruiz Silva | Bellas Vistas | Terapeuta |
| Matilde | Martínez Andrés | Bellas Vistas | Terapeuta |
| Rocío | Mir Méndez | Bellas Vistas | Terapeuta |

Mostrando 1 a 10 de 11 resultados Anterior 1 2 Siguiente

*Figura 39. Actualización tabla consulta. Prueba de baja del sistema*

## 5.2.Pruebas relacionadas con la usabilidad del sistema

En este apartado se pretenden describir y validar distintas técnicas utilizadas para ofrecer un sistema usable a los usuarios. Estas técnicas se basan en facilidades que se ofrecen en la aplicación web para que la navegación sea mucho más intuitiva, sencilla y útil.

### 5.2.1. Visualización de menús en pestañas

Uno de los aspectos de diseño requeridos para una mayor usabilidad de la página web, consiste en añadir un menú de pestañas que ofrezca un rápido acceso a las funcionalidades que el sistema ofrece en la página de bienvenida, desde cualquier menú. De esta forma el usuario sin necesidad de regresar a la página de bienvenida, puede acceder cómodamente a cualquiera de los menús. Se ha conseguido implementar un menú de pestañas para cada página .html, y se ha comprobado que las pestañas responden correctamente.

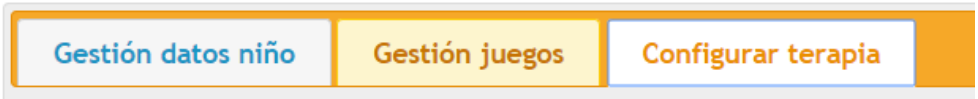


*Figura 40. Pestañas usuarios médicos*


*Figura 41. Pestañas usuario administrador*

Las pestañas permiten moverse de forma sencilla por los menús del sistema. La pestaña recuadrada en azul con la letra naranja corresponde a la pestaña activa, si pasamos el ratón por encima de alguna de ellas aparece una sombra naranja como puede verse en las Figuras 40 y 41. Las pestañas inactivas se muestran en otro color en este caso azul, como puede verse en la Figura 40.

### 5.2.2. Paginación del contenido de las tablas

Los menús de consulta pueden ofrecer una gran cantidad de datos a los usuarios. Para facilitar su consulta, todas las tablas implementan un mecanismo de paginación que nos permite visualizar en cada página 10, 25, 50 o 100 resultados. De forma, como el caso que se refleja en las siguientes figuras, los resultados pueden visualizarse repartidos en páginas sucesivas o todos en la misma página. Se ha comprobado que la paginación funciona correctamente, asegurando así una mayor comodidad en la consulta de datos. Este resultado puede observarse en las Figuras 42, 43 y 44.


Listado de profesionales

Mostrando 10 resultados

| Nombre | Apellidos | Centro | Especialidad |
|---|---|---|---|
| Alberto | Cortázar Castro | Bellas Vistas | Terapeuta |
| Belén | Duro González | Bellas Vistas | Terapeuta |
| Carlos | González Grande | Bellas Vistas | Terapeuta |
| Elena | Rodríguez Sanz | Bellas Vistas | Terapeuta |
| Francisco | Gutiérrez Moreno | Bellas Vistas | Terapeuta |
| Inma | Abella Villa | Bellas Vistas | Terapeuta |
| Juan | Herrera Vidal | Bellas Vistas | Terapeuta |
| Laura | Sánchez García | Bellas Vistas | Terapeuta |
| María | Ruiz Silva | Bellas Vistas | Terapeuta |
| Matilde | Martínez Andrés | Bellas Vistas | Terapeuta |

Mostrando 1 a 10 de 12 resultados Anterior 1 2 Siguiente

*Figura 42. Paginación*

| Nombre | Apellidos | Centro | Especialidad |
|---|---|---|---|
| Rocío | Mir Méndez | Bellas Vistas | Terapeuta |
| Rodrigo | Agudo Fernández | Bellas Vistas | Terapeuta |

Mostrando 11 a 12 de 12 resultados  Anterior 1 2 Siguiente

*Figura 43. Paginación 2*

Listado de profesionales

Mostrando 25 resultados

| Nombre | Apellidos | Centro | Especialidad |
|---|---|---|---|
| Alberto | Cortázar Castro | Bellas Vistas | Terapeuta |
| Belén | Duro González | Bellas Vistas | Terapeuta |
| Carlos | González Grande | Bellas Vistas | Terapeuta |
| Elena | Rodríguez Sanz | Bellas Vistas | Terapeuta |
| Francisco | Gutiérrez Moreno | Bellas Vistas | Terapeuta |
| Inma | Abella Villa | Bellas Vistas | Terapeuta |
| Juan | Herrera Vidal | Bellas Vistas | Terapeuta |
| Laura | Sánchez García | Bellas Vistas | Terapeuta |
| María | Ruiz Silva | Bellas Vistas | Terapeuta |
| Matilde | Martínez Andrés | Bellas Vistas | Terapeuta |
| Rocío | Mir Méndez | Bellas Vistas | Terapeuta |
| Rodrigo | Agudo Fernández | Bellas Vistas | Terapeuta |

Mostrando 1 a 12 de 12 resultados  Anterior 1 Siguiente

*Figura 44. Paginación 3*

### 5.2.3. Aplicación de distintos criterios de ordenación.

Además de las facilidades comentadas, para realizar una consulta más eficiente, se ha decidido implementar una tabla que permita ordenar sus datos bajo cualquier criterio de consulta de los disponibles en la cabecera de la tabla. Para ello, deberemos hacer clic en la descripción del campo según el cual queremos ordenar. Además, tras un segundo clic en el mismo campo, se mantiene el criterio de ordenación pero en orden descendente.

Por ejemplo, en la Figura 32, los datos niños se ordenaban según sus apellidos de manera ascendente. Ahora se muestran ejemplos de ordenación según la edad (Figura 45)

y el nombre (Figura 46). Al hacer un clic la ordenación obtenemos una ordenación ascendente según el criterio seleccionado como puede verse en las figuras. Si volvemos a hacer clic en el mismo campo, la ordenación se realiza en base al mismo criterio pero de manera descendente como puede verse en la Figura 47.

| Nombre | Apellidos | Fecha de nacimiento | Edad |
|---|---|---|---|
| Laura | Martínez Gil | 10-10-2010 | 4 |
| Javier | García García | 16-07-2009 | 6 |
| Manuel | Pérez Martín | 01-12-2007 | 7 |
| Belén | González González | 02-02-2006 | 9 |
| Miriam | Muñoz Díaz | 25-5-2005 | 10 |

*Figura 45. Consulta de niños ordenada por edad*

| Nombre | Apellidos | Fecha de nacimiento | Edad |
|---|---|---|---|
| Belén | González González | 02-02-2006 | 9 |
| Javier | García García | 16-07-2009 | 6 |
| Laura | Martínez Gil | 10-10-2010 | 4 |
| Manuel | Pérez Martín | 01-12-2007 | 7 |
| Miriam | Muñoz Díaz | 25-5-2005 | 10 |

*Figura 46. Consulta de niños ordenada por nombre*

| Nombre | Apellidos | Fecha de nacimiento | Edad |
|---|---|---|---|
| Miriam | Muñoz Díaz | 25-5-2005 | 10 |
| Belén | González González | 02-02-2006 | 9 |
| Manuel | Pérez Martín | 01-12-2007 | 7 |
| Javier | García García | 16-07-2009 | 6 |
| Laura | Martínez Gil | 10-10-2010 | 4 |

*Figura 47. Consulta de niños por edad descendente*

Este mecanismo de ordenación tan sencillo puede utilizarse en todas las tablas de la aplicación web para una mayor comodidad en la visualización de los resultados o en la búsqueda de algún registro concreto.

### 5.2.4. Mensajes de error

Se ha comentado a lo largo de este documento, el tratamiento de errores que realiza el WS en cuanto al problema de la replicación de claves en la base de datos. El sistema asegura que no se introduzcan claves primarias repetidas en la base de datos, sin embargo, es necesario avisar al usuario de este problema, para que pueda introducir su registro en el sistema, modificando el valor afectado del formulario. Para ello, el sistema muestra un mensaje de error en el propio formulario, indicando qué campo debe modificar.

De esta manera aseguramos una realimentación hacia el usuario por parte del sistema que permita al usuario introducir el registro correctamente. Se muestra a continuación un ejemplo de replicación de claves al intentar dar de alta un niño en el sistema. El identificador introducido ya existía en el sistema, por tanto, se informa al usuario de la necesidad de modificar este valor para poder dar de alta al niño (Figura 48). Este problema se ha gestionado de la misma forma en otros formularios, como el de alta de profesionales médicos.


*Figura 48. Mensaje de error clave primaria repetida*

# 6. Conclusiones y trabajos futuros

## 6.1.Conclusiones

El uso de nuevas tecnologías en el ámbito de la telemedicina y la e-salud está ofreciendo avances muy significativos en este contexto, sobre todo, cuando se trata de niños. Manteniendo el objetivo de ofrecer sistemas visualmente atractivos para ellos, se ha diseñado una plataforma web cuyo componente más visual, la interfaz, debía ajustarse a este objetivo, lo que ha condicionado totalmente su diseño en colores vivos y alegres y el uso de tecnologías muy novedosas para su implementación.

El objetivo principal de este proyecto era permitir a los especialistas el acceso a la configuración de los juegos y los usuarios del sistema SONRIE. Este hecho se ha tenido muy en cuenta también en el diseño de la interfaz, asegurando su usabilidad y sencillez. Además, se ha valorado el conjunto de usuarios que utilizarán el sistema, a la hora de diseñar una lógica que evite errores derivados de escrituras incorrectas o repetición de claves del sistema. No olvidemos que los principales beneficiarios de este sistema son los niños. Este beneficio se materializa en forma de un mayor seguimiento de su terapia y la posibilidad de influir sobre ella para tratar de mejorar la ejecución de ciertos movimientos faciales. Todo ello se ha conseguido ofrecer de una forma muy sencilla e intuitiva a los profesionales que los atienden.

La plataforma desarrollada, utiliza tecnologías web, asegurando que se pueda acceder a ella en cualquier momento, dada la enorme variedad de dispositivos móviles con acceso a internet que se encuentran disponibles en el mercado y que ya forman parte de la sociedad. Sin embargo, ofrece seguridad en el acceso, limitándolo únicamente a usuarios del sistema SONRIE.

La lógica de negocio sigue el modelo REST, que se ajusta perfectamente al entorno en el que se sitúa el sistema ya que, debemos asumir muy pocas suposiciones sobre las capacidades tecnológicas del receptor. Una lógica de negocio de tipo SOAP hubiera complicado enormemente la implementación del lado del cliente, y, en caso de inclusión de nuevas funcionalidades del sistema, este tipo de implementación hubiera condicionado crear nuevamente las clases del cliente teniendo en cuenta estas modificaciones. La lógica generada en el lado del cliente, sería mucho más pesada que la lógica implementada en este proyecto y además, con el empleo de esta tecnología, conseguimos ofrecer la posibilidad de escalar la aplicación rápidamente reutilizando el código desarrollado.

El tipo de WS generado, así como, la gran variedad de widgets disponibles ha condicionado el uso de JQuery. Este hecho ha permitido implementar una interfaz muy vistosa y con un rápido acceso al resto de componentes y por tanto, que proporciona un rápido acceso y visualización de la información del sistema. El uso de otras tecnologías más antiguas no hubiese permitido ofrecer una interfaz tan dinámica y vistosa.

La arquitectura de componentes desarrollada fomenta la reutilización de código. Esta facilidad, que ya se ha utilizado en el desarrollo de la solución, permitirá añadir nuevas funcionalidades al sistema de una forma muy sencilla y rápida.

La documentación del proyecto dentro del propio código, en forma de comentarios, o accesible a través de la documentación de la API desarrollada, ofrece el soporte fundamental para futuras evoluciones del sistema.

Como conclusión final, el producto software desarrollado es una herramienta totalmente validada, usable y totalmente basada en nuevas tecnologías, que permite cubrir las necesidades de los actores que intervienen en el seguimiento terapéutico de los niños con PCI. Además, la arquitectura de componentes desarrollada permite una evolución del mismo de manera sencilla.

La implementación cubre perfectamente los requisitos y objetivos del proyecto, sin embargo, la limitación del tiempo disponible para realizar el proyecto, ha condicionado que ciertos aspectos que mejorarían el resultado final no hayan podido implementarse. Es por ello, que se enumeran a continuación las posibles mejoras que pueden incluirse en el sistema en su futura evolución.

## 6.2.Trabajos futuros

A continuación se enumeran algunas de las futuras mejoras que pueden introducirse en el sistema desarrollado:

- Añadir funcionalidades relativas a la gestión de centros y de especialidades en el usuario administrador del sistema, que incluyan menús de consulta, modificación y alta y baja de centros y especialidades médicas.
- Asegurar la interoperabilidad entre este framework y la plataforma de juegos SONRIE [SAM15] desarrollada por Estefanía Sampedro. En primer lugar, deberá asegurarse que la información que la plataforma de juegos recibe del niño y la información que el sistema genera en base a la anterior, se almacenen correctamente en la BD del sistema. Además, la plataforma de juegos deberá iniciarse, para cada niño, con la última configuración almacenada en el sistema para cada juego. Estas configuraciones establecidas por los propios especialistas a través de la aplicación web, son de vital importancia para asegurar un correcto seguimiento de la terapia de los niños, así como, una total adecuación de la plataforma de juegos a cada niño, en función de las acciones llevadas a cabo por los especialistas.
- Añadir la posibilidad de consultar estadísticamente los resultados obtenidos tras la realización de los juegos según distintos criterios.

- Asegurar la correcta disposición de los distintos elementos de la página web en cualquier formato de pantalla.
- Con la experiencia y conocimientos adquiridos en el desarrollo de este framework podrán realizarse otras aplicaciones web adaptadas a contextos más allá de la telemedicina. Se ha conseguido enlazar un conjunto bastante grande de componentes basados en nuevas tecnología, obteniendo un desarrollo totalmente extrapolable a cualquier área en la que se necesite un almacenaje persistente de la información y una gestión activa de ésta.

# 7. Bibliografía

**[APA15]** The Apache Software Foundation Apache Tomcat, http://tomcat.apache.org/index.html

**[BER93]** Berenson, M. Estadística para administración y economía. MCGraw-Hill; 1993. p. 410-496.

**[CHE14]** Chen YP, Lee SY, Howard AM. Effect of virtual reality on upper extremity function in children with cerebral palsy: a meta-analysis. PediatrPhysTher., 2014

**[ECM13]** ECMA-404 Standard, The JSON Data Interchange Format, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

**[GOO15]** Google, Project HTML5 Rocks, HTML5 features http://www.html5rocks.com/en/features/semantics

**[KEN14]** Kenkov J. , Web Service Message Formats http://tutorials.jenkov.com/web-services/message-formats.html

**[LIB15]** LibrosWeb, AJAX, https://librosweb.es/libro/ajax/

**[LUN13]** Luna-Oliva L, Ortiz-Gutiérrez RM, Cano-de la Cuerda R, Piédrola RM, Alguacil-Diego IM, Sánchez-Camarero C, Martínez Culebras MC. Kinect Xbox 360 as a therapeutic modality for children with cerebral palsy in a school environment: a preliminary study. NeuroRehabilitation, 2013

**[MON15]** Mono Project, http://www.mono-project.com/

**[ORA13]** Oracle, The Java EE 6 tutorial, Types of Web services, https://docs.oracle.com/javaee/6/tutorial/doc/giqsx.html
Oracle, The Java EE 6 tutorial, REST Web Service, http://docs.oracle.com/javaee/6/tutorial/doc/gilik.html

**[ORA14]** Oracle, Java Platform, Enterprise Edition The Java EE Tutorial, Release 7, https://docs.oracle.com/javaee/7/JEETT.pdf

**[ORA15]** Oracle, MySQL, http://www.oracle.com/es/products/mysql/overview/index.html
Oracle, Transacciones, https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html

**[PRE]** Paul Prescod, "Roots of the REST/SOAP Debate". http://www.prescod.net/rest/rest_vs_soap_overview/

**[SAM15]** Sampedro Estefanía, Sistema de apoyo terapéutico con Kinect para niños con parálisis cerebral infantil.

**[SHA12]** Sharan D, Ajeesh PS, Rameshkumar R, Mathankumar M, Paulina RJ, Manjula M. Virtual reality based therapy for postoperative rehabilitation of children with cerebral palsy, 2012.

**[SUN01]** Sun Microsystems, Data Access Object http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

**[W3C14]** W3C, HTML5 Recommendation, http://www.w3.org/TR/html5/

**[W3C15]** W3C, Guía breve de servicios web, http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

**[W3S]** W3schools,http://www.w3schools.com/

**[WAN11]** Wang M, Reid D. Virtual reality in pediatric neurorehabilitation: attention deficit hyperactivity disorder, autism and cerebral palsy. Neuroepidemiology, 2011.

**[WIK15]** Wikipedia: La enciclopedia libre
https://es.wikipedia.org/wiki/JSON
https://es.wikipedia.org/wiki/Gson
Consulta: Mayo 2015

# Anexo A: Manual de usuario del Sistema SONRIE

La plataforma de juegos comienza con la siguiente pantalla inicial (Figura 1)


*Figura 49. Pantalla inicial*

A continuación, aparecen las niñas que serán las guías durante todo el desarrollo del juego (Figura 2). El niño debe estar colocado frente a la cámara.

*Figura 2. Presentación de las guías de juego*

Cuando las niñas digan "¡Hola!" el niño debe saludarlas con la mano Una vez haya finalizado el vídeo, se mostrará la siguiente pantalla donde comienza el primer juego, el juego de las cejas. El objetivo es que el niño sea capaz de levantar las cejas. Una de las niñas explica cómo se debe realizar el movimiento (Figura 3).

*Figura 3. Juego de las cejas*

Si el niño realiza el movimiento de manera correcta, o en su defecto, se sobrepasa el tiempo límite establecido para cada intento de este juego, se muestra la imagen con las cortinas descorridas y una mesa con el fondo de unos aplausos (Figura 4). Si este juego se ha ejecutado un número de veces inferior al número de intentos establecido, se volverá al primer vídeo.

*Figura 4. Final del juego de las cejas*

Cuando el juego se ejecuta el número de veces que se haya asignado (por defecto son tres veces), comienza el siguiente, el juego del soplido. En este juego el niño debe soplar como indica la niña del vídeo (Figura 5), para apagar la vela que se encuentra encendida encima de la mesa.

*Figura 5. Juego del soplido*

Si se ejecuta el movimiento o se excede el límite de tiempo aparece la siguiente imagen, en la que se muestra la vela apagada y se reproducen unos aplausos (Figura 6) Este vídeo se repetirá hasta que se reconozca el soplido, o se sobrepase el tiempo límite. Entonces, se mostrará la misma imagen, pero sin la niña situada en la parte inferior y con la llama de la vela apagada mientras se reproducen unos aplausos, como se muestra en la Figura 6. Si el juego se realiza el número de veces asignado, por defecto tres, se da paso al siguiente juego. En caso contrario, se volverá al vídeo con la vela encendida indicando que el niño debe ejecutar nuevamente el movimiento.

*Figura 6. Fin del juego del soplido*

Una vez que el niño ha superado la anterior pantalla, el juego del soplido, se da paso al siguiente desafío, en este caso el objetivo es lanzar un beso. Se muestra en pantalla las tres niñas, guías del juego, con el mismo fondo de las cortinas con la mesa, simulando estar enfadadas, mientras que una de ellas aparece en la parte inferior derecha explicando cómo debe realizarse el movimiento (Figura 7).

*Figura 7. Juego del beso*

Cuando el niño lo ejecuta correctamente o se excede el límite de tiempo, las niñas sonríen (Figura 8). Si el juego se ejecuta tres veces se pasa al cuarto juego, el de la sonrisa (Figura 9)

*Figura 8. Fin juego del beso*

El cuarto y último de los juegos consiste en que el niño sonría, para ello aparece sobre el mismo fondo, una de las niñas indicándole que debe sonreír como está haciendo ella para finalizar el juego (Figura 9).

*Figura 9. Juego de la sonrisa*

Una vez se haya ejecutado el movimiento el número de veces asignado, que por defecto corresponde a tres veces, aparece la pantalla final. Esta pantalla no necesita la interacción del niño, sin embargo, las niñas le informan de que es el final del juego y que lo ha realizado correctamente mientras saltan y se abrazan (Figura 10)

*Figura 10. Pantalla final*

# Anexo B: Manual de usuario de la plataforma desarrollada


## PLATAFORMA TERAPEÚTICA BASADA EN KINECT PARA NIÑOS CON PARÁLISIS CEREBRAL INFANTIL


*Figura 1. Página de entrada al sistema SONRIE*


*Figura 2. Página de entrada al sistema SONRIE como administrador*

# 1. Entrar al sistema como usuario médico

La página de entrada al sistema, representada a continuación, estará accesible a través de Internet en la dirección http://marisa.diatel.upm.es. Para mayor detalle, también se muestra la imagen ampliada en la Figura 1.


*Figura 3. Página inicial*

Los profesionales, para acceder a las funcionalidades que el sistema ofrece, deberán iniciar sesión introduciendo su nombre de usuario y contraseña en el menú de autenticación que se indica en la Figura 4, y pulsando a continuación el botón *Enviar*.


*Figura 4. Autenticación de entrada al sistema SONRIE*

Si se introduce un nombre de usuario que no existe o una contraseña errónea se muestran los siguientes mensajes de error:

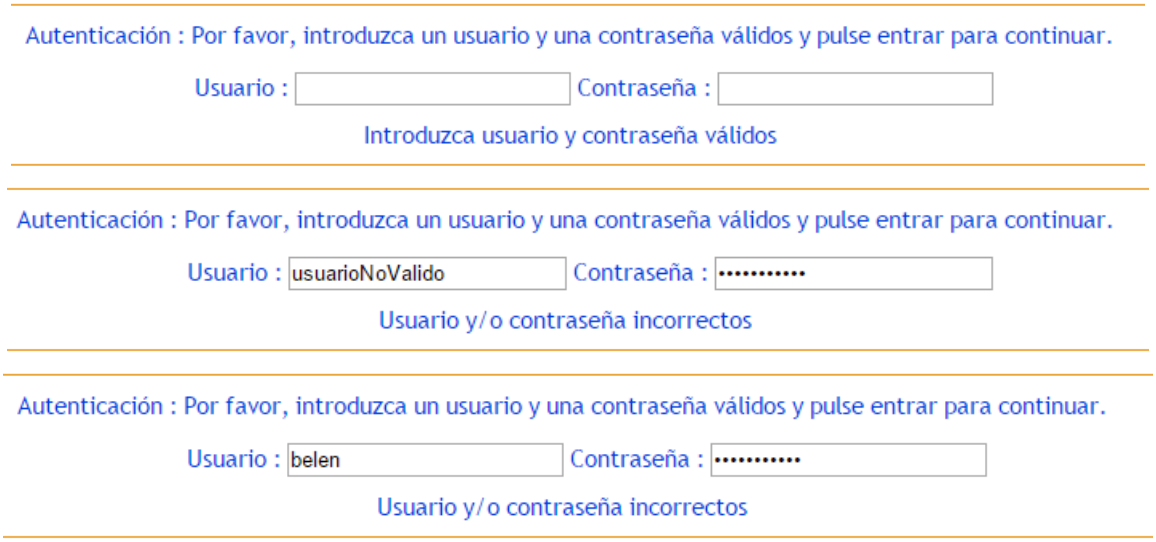


*Figura 5. Errores en el inicio de sesión*

Tenga especial atención si los errores se repiten pues el sistema distingue entre mayúsculas y minúsculas en este menú. En caso de no recordar sus credenciales de usuario deberá ponerse en contacto con el administrador del sistema para que se las proporcione.

Una vez introducidas las credenciales correctas, tendrá acceso a las funcionalidades del sistema a través de la página de bienvenida.


*Figura 6. Página de bienvenida*

Desde esta página podrá tener acceso a la gestión de los niños a los que atiende, así como, a la gestión de sus terapias. También podrá consultar los juegos del sistema. Estas funcionalidades se describen con mayor detalle a continuación. Además, si lo desea podrá cerrar sesión con el botón situado en la esquina superior izquierda.

## 2. Gestión datos niño

Si pulsa el botón *Gestión datos niño* del menú principal, se mostrará una tabla con los niños del sistema a los que usted atiende informando de sus nombres y apellidos, así como, de su fecha de nacimiento y edad.

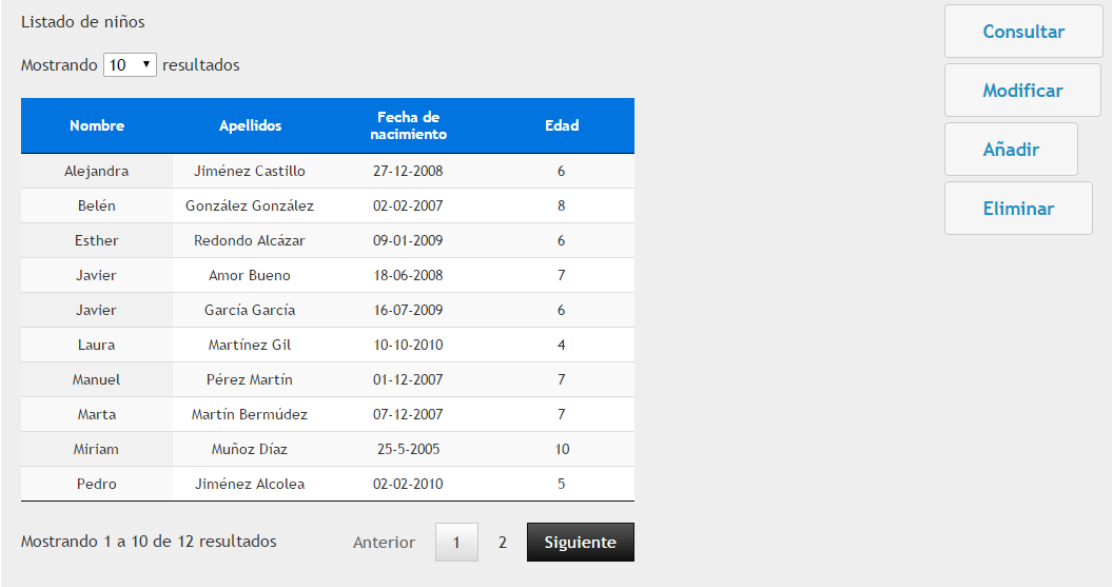


*Figura 7. Consulta de niños*

Si desea volver a esta pantalla desde cualquier otra opción, pulse el botón *Consultar*. Si se encuentra en otro menú al pulse la pestaña *Gestión de datos niño* del menú de pestañas (Figura 8), para visualizar esta pantalla. En el menú de pestañas, la pestaña actual aparece con letras naranjas recuadrada en azul. El resto de pestañas, las inactivas, se mostrarán en color azul. Si pasa el ratón por alguna de las pestañas aparecerá una sombra color naranja.

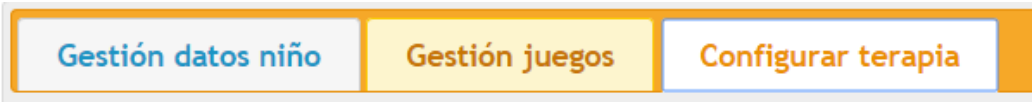


*Figura 8. Menú de pestañas*

En el menú de gestión de consulta de niños, los resultados se muestran en distintas páginas a las que puede acceder según le interese. Para ello, puede utilizar los botones o las opciones *Anterior* y *Siguiente* que se muestran en la parte inferior de la tabla y en la Figura 9. En la parte inferior aparece el número de niños que se está visualizando en esta tabla, junto con el total de niños a los que usted atiende. Si desea modificar el número de resultados mostrados en cada página, utilice el menú se selección que aparece en la parte superior de la tabla. En la Figura 11 se muestra un ejemplo de esta modificación del número de resultados utilizando este menú de selección.


*Figura 9. Botones paginación*

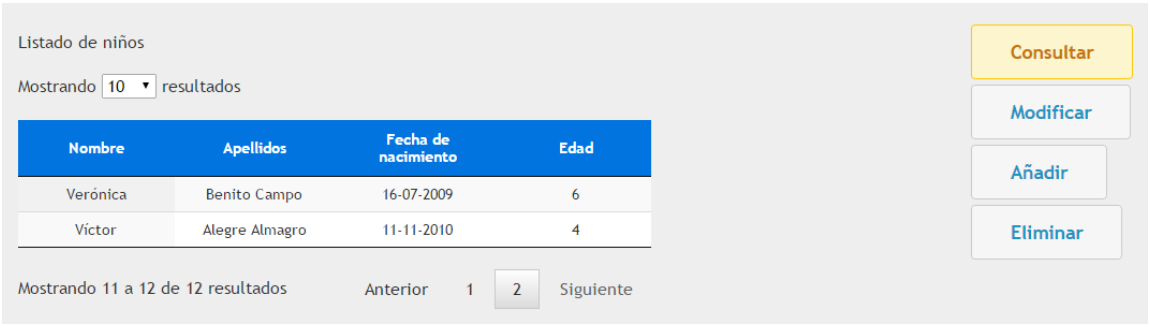


*Figura 10. Consulta de niños. Página siguiente*

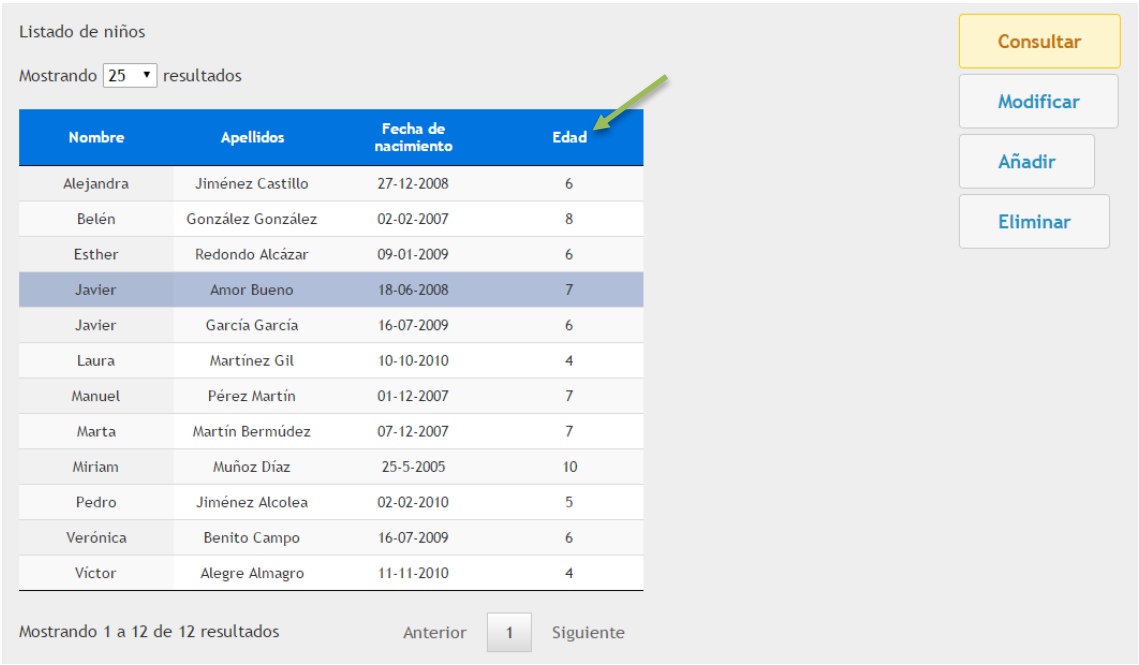


*Figura 11. Modificación del menú de selección y ordenación de datos*

Además si lo desea, podrá ordenar la tabla pulsando en la cabecera de esta el criterio de ordenación que quiere aplicar. Por ejemplo, si desea ordenar el conjunto de niños por su edad, deberá pulsar en Edad como se ha indicado en la Figura 11. Pulse de nuevo si desea que la ordenación sea de mayor a menor. Esta funcionalidad será muy útil si desea localizar a alguno de los niños.

Deberá utilizar la tabla de consulta, mostrada en este apartado, para seleccionar el niño que desea modificar o dar de baja del sistema. Pulse el botón *Consultar*, la pestaña *Gestión datos niño,* o el botón *Gestión datos niño* de la página de bienvenida antes de seleccionar la fila que desea modificar o eliminar.

## 2.1.Dar de alta un niño en el sistema

Si desea dar de alta un nuevo niño en el sistema diríjase a la tabla consulta del menú gestión de niños utilizando el botón *Consultar* o la pestaña *Gestión datos niño* y asegúrese

de que ninguna fila de la tabla se encuentra seleccionada, pulsando en el espacio disponible entra la tabla y los datos. Y a continuación pulse el botón *Añadir*.


*Figura 12. Dar de alta niño*

Se mostrará un menú (Figura 12) en el que debe introducir todos los datos del niño a excepción de la dirección que es un campo opcional. Observará que en este menú aparecen su nombre y apellidos como profesional que atiende al niño de cara al sistema. No debe modificar estos datos.

Una vez haya introducido los datos del niño que desea dar de alta, pulse el botón *Enviar.* A continuación se mostrará la tabla de consulta en la que podrá visualizar los datos del niño dado de alta en el sistema.

## 2.2.Modificar datos de un niño

Si desea modificar los datos de un niño registrado en el sistema diríjase a la tabla consulta del menú gestión de niños utilizando el botón *Consultar* o la pestaña *Gestión datos* niño. Seleccione la fila que corresponde a los datos del niño que desea modificar, y a continuación, pulse el botón *Modificar*. Aparecerá un formulario (Figura 13) en el que podrá visualizar los datos del niño seleccionado.

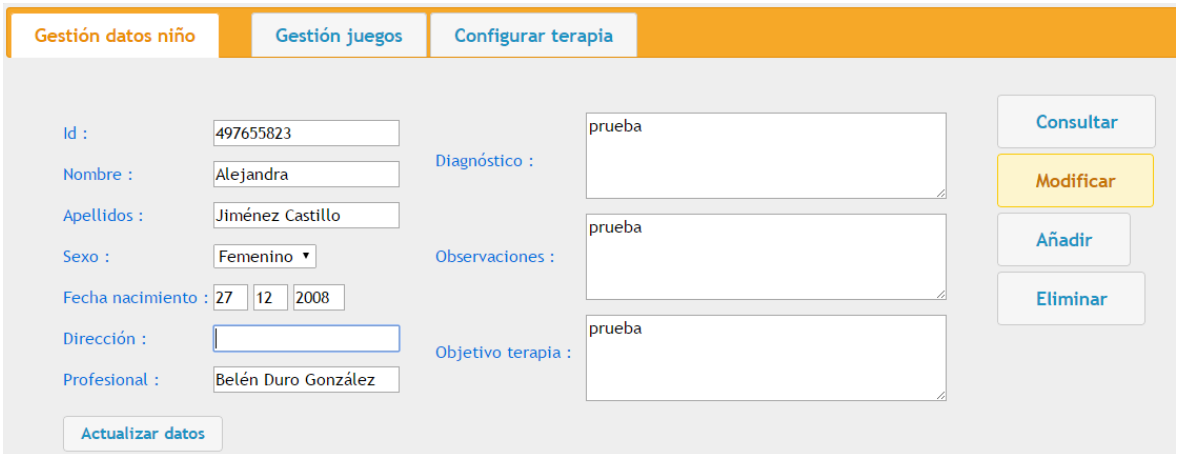


*Figura 13. Modificar datos de niño*

Si modifica estos datos y pulsa el botón *Actualizar datos*, se cargarán estos nuevos valores en el sistema.

### 2.3.Dar de baja a un niño del sistema

Si desea dar de baja a alguno de los niños del sistema, deberá regresar a la tabla de consulta de niños a través del botón *Consultar* o la pestaña de *Gestión de datos niño*.

Una vez visualice la tabla podrá seleccionar el niño que desea dar de baja del sistema. Si la tabla no permite seleccionar pulse de nuevo el botón Consultar. Una vez seleccionado si pulsa el botón *Eliminar* se dará de baja el niño seleccionado. A continuación, se volverá a mostrar la tabla de consulta en la que podrá observar que la fila perteneciente al niño que acaba de ser dado de baja, ya no existe.

## 3. Gestión de juegos

Si pulsa el botón *Gestión juegos* del menú principal o la pestaña *Gestión juegos*, se mostrará una tabla con los juegos disponibles en el sistema junto con su objetivo.

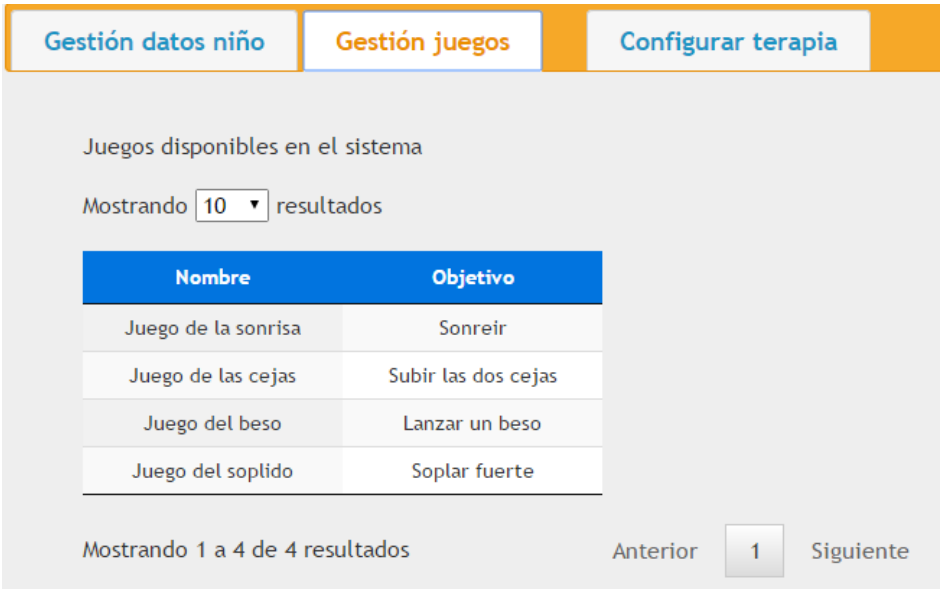


*Figura 14. Consulta de juegos*

## 4. Configurar terapia

Si pulsa el botón *Configurar terapia* del menú principal o la pestaña *Configurar terapia,* podrá visualizar el conjunto de los niños a los que atiende (Figura 15). Utilice los botones o las opciones *Anterior* y *Siguiente* que se muestran en la parte inferior de la tabla para moverse por las distintas páginas.

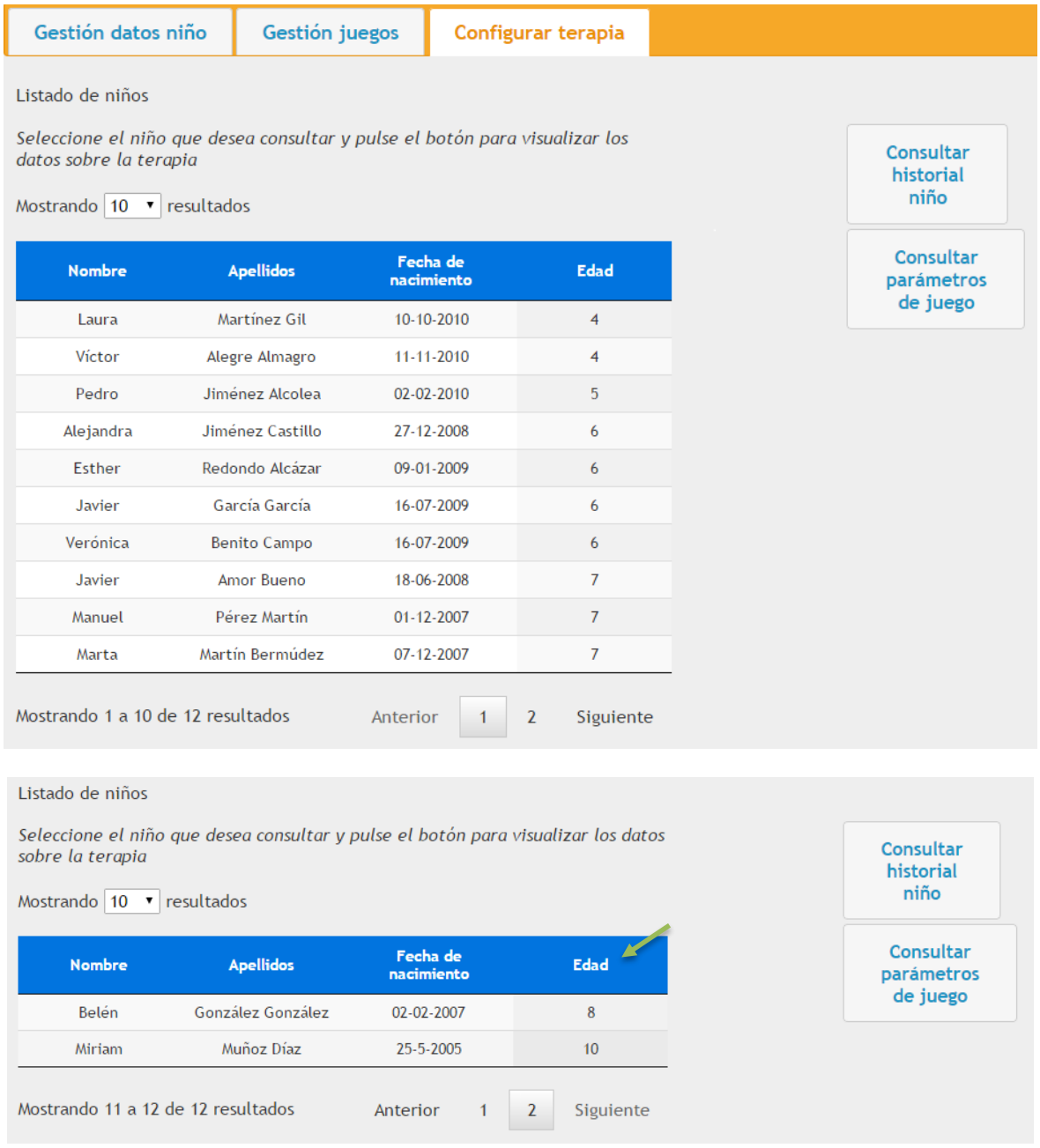


*Figura 15. Tabla niños*

Como podrá observar, los resultados se encuentran ordenados según la edad. Para ordenar los resultados bajo cualquier criterio de la cabecera de la tabla pulse en la columna correspondiente. En este caso se pulsado en Edad, como señala la flecha de la Figura 15. Si desea ordenar de manera descendente, esto es, de mayor a menor, vuelva a pulsar en el criterio.

Si desea modificar el número de resultados mostrados en cada página, utilice el menú se selección que aparece en la parte superior de la tabla. (Figura 16)


*Figura 16. Tabla niños cambio número de resultados mostrados*

Para consultar el historial de alguno de los niños, selecciónelo y pulse el botón *Consultar historial niño*. (Figura 17)

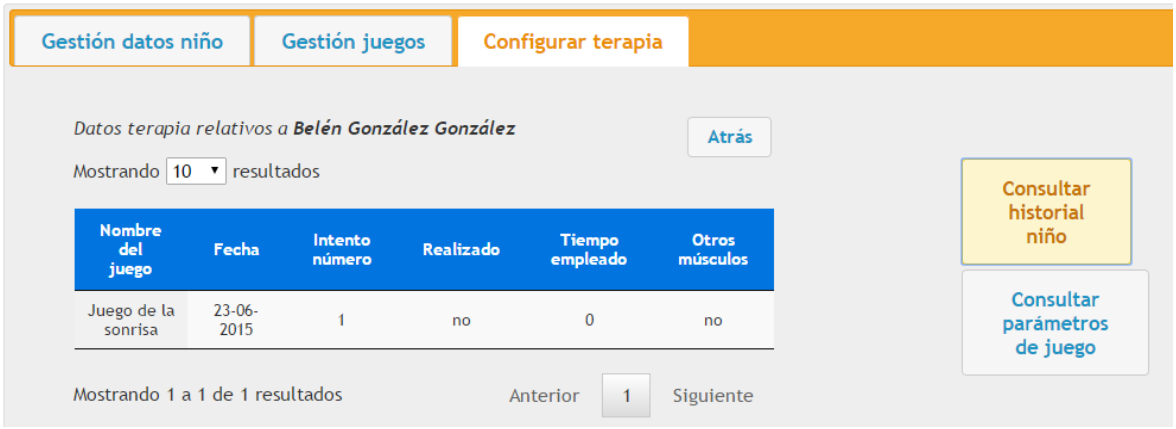


*Figura 17. Consulta de historial de un niño*

En este menú, se mostrarán los datos registrados acerca de los juegos que ha realizado el niño. Pulse el botón atrás cuando haya finalizado su consulta.

Regresará a la tabla de consulta inicial. Si desea consultar las configuraciones de los juegos (el número de intentos y el tiempo límite para cada intento de cada juego) añadidas para alguno de los niños, selecciónelo en la tabla y pulse el botón *Consultar parámetros de juego.*

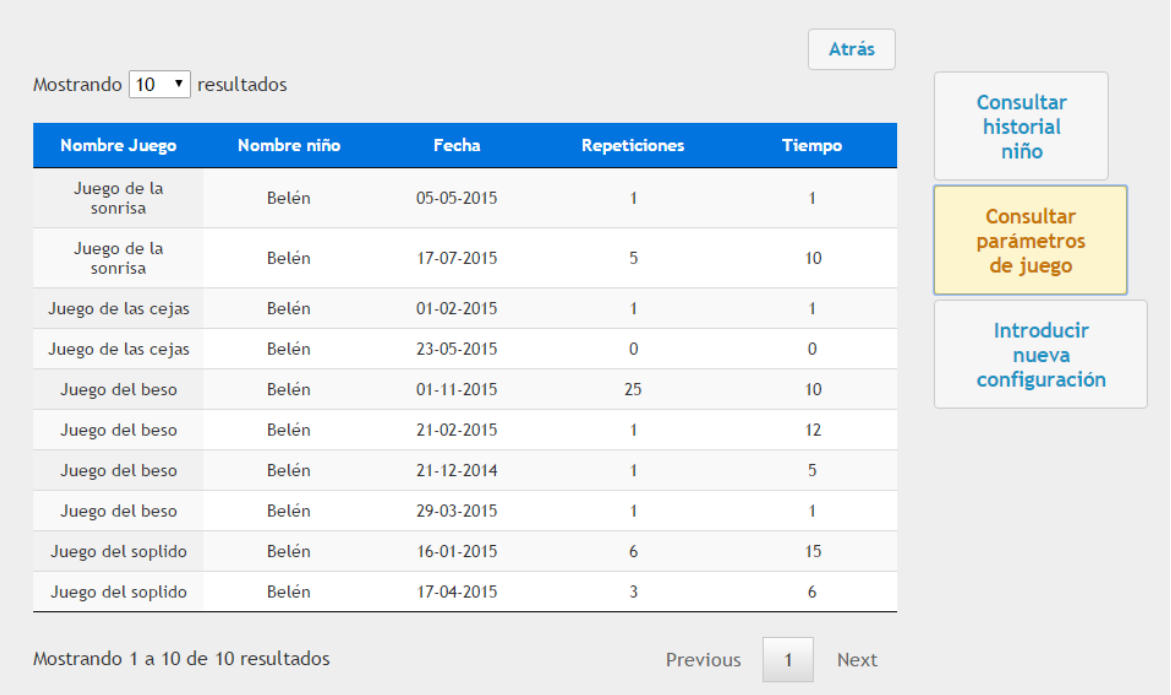


*Figura 18. Consultar parámetros de juego*

En la consulta de parámetros de juego podrá visualizar todas las configuraciones añadidas al sistema para este niño. Si desea añadir una nueva configuración pulse el botón *Introducir nueva configuración*. Y continuación se mostrará un menú en el que debe seleccionar el juego, introducir la fecha actual e indicar el valor de los nuevos parámetros del juego para el niño seleccionado: el número de repeticiones y el tiempo límite en segundos para cada uno de los intentos (Figura 19)

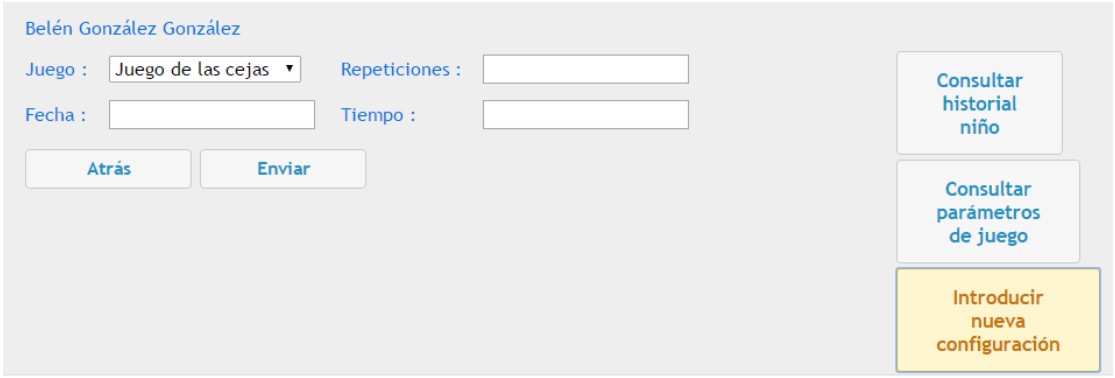


*Figura 19. Introducir una nueva configuración*

Pulsando el botón atrás podrá regresar a la tabla de consulta del menú Configurar terapia.

## 5. Volver a la página de bienvenida

Cuando se encuentre en cualquier página, podrá regresar a la página de bienvenida, de manera sencilla, pulsando el logo del sistema SONRIE tal y como se muestra en la Figura 20


*Figura 20. Volver a la página de inicio*

## 6. Cerrar sesión

Una vez haya completado sus gestiones podrá cerrar sesión en el sistema pulsando el botón *Cerrar sesión* disponible en cualquier página de la aplicación. Entonces, aparecerá la página de entrada al sistema con el menú de autenticación.


*Figura 21. Cerrar sesión*

## 7. Entrar como administrador

Como usuario administrador del sistema dispone de una serie de funcionalidades para realizar una gestión eficiente de profesionales y juegos del sistema. Para acceder a estas opciones en primer lugar deberá entrar en el acceso administrativo disponible en la página de entrada al sistema SONRIE indicado en la Figura 22.


*Figura 22. Acceso administrativo al sistema*

La página del administrador se muestra en la Figura 23 y en mayor detalle en la Figura 2 situada al comienzo de este Anexo. Una vez acceda a página de entrada del acceso administrativo, deberá introducir la contraseña del usuario administrador en el menú de autenticación (Figura 24) para poder acceder a las opciones que el sistema le ofrece.


*Figura 23. Menú autenticación del acceso administrativo*


*Figura 24. Menú autenticación del acceso administrativo*

Al introducir la contraseña correcta se habilitarán las opciones de gestión de profesionales y gestión de juegos del sistema que se detallan en los siguientes apartados y que pueden visualizarse en la siguiente figura.


*Figura 25. Página de bienvenida acceso administrativo*

# 8. Gestión de profesionales

Una vez haya pulsado el botón *Gestión de profesionales* del menú principal se mostrará una tabla con los profesionales dados de alta en el sistema (Figura 26).

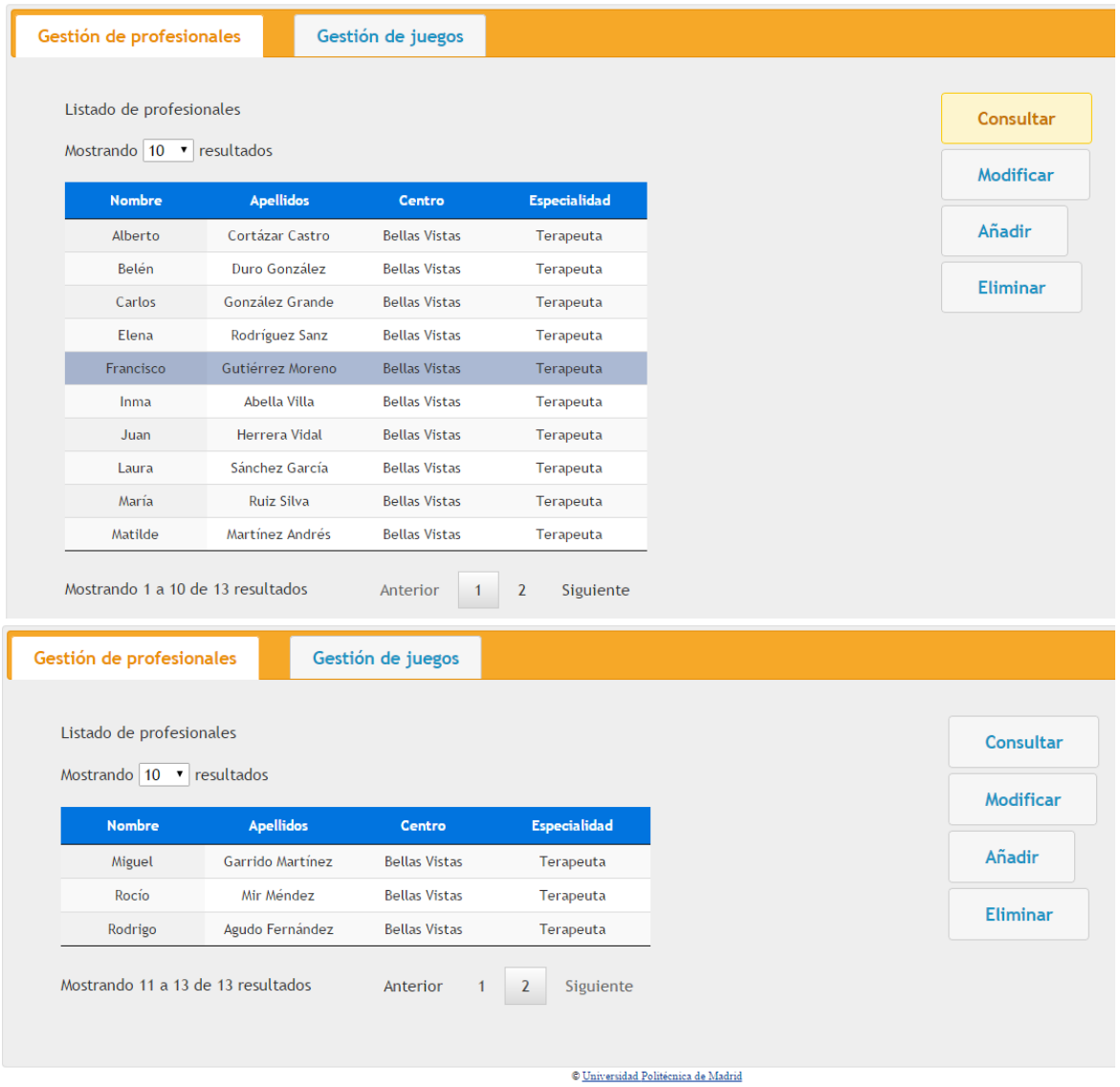


*Figura 26. Consulta de usuarios médicos*

Podrá utilizar los botones mostrados en la Figura 27 para moverse por las distintas páginas. Si desea obtener un número distinto de resultados en cada página, modifique el valor del menú de selección de resultados, situado encima de la tabla.


*Figura 27. Botones paginación*

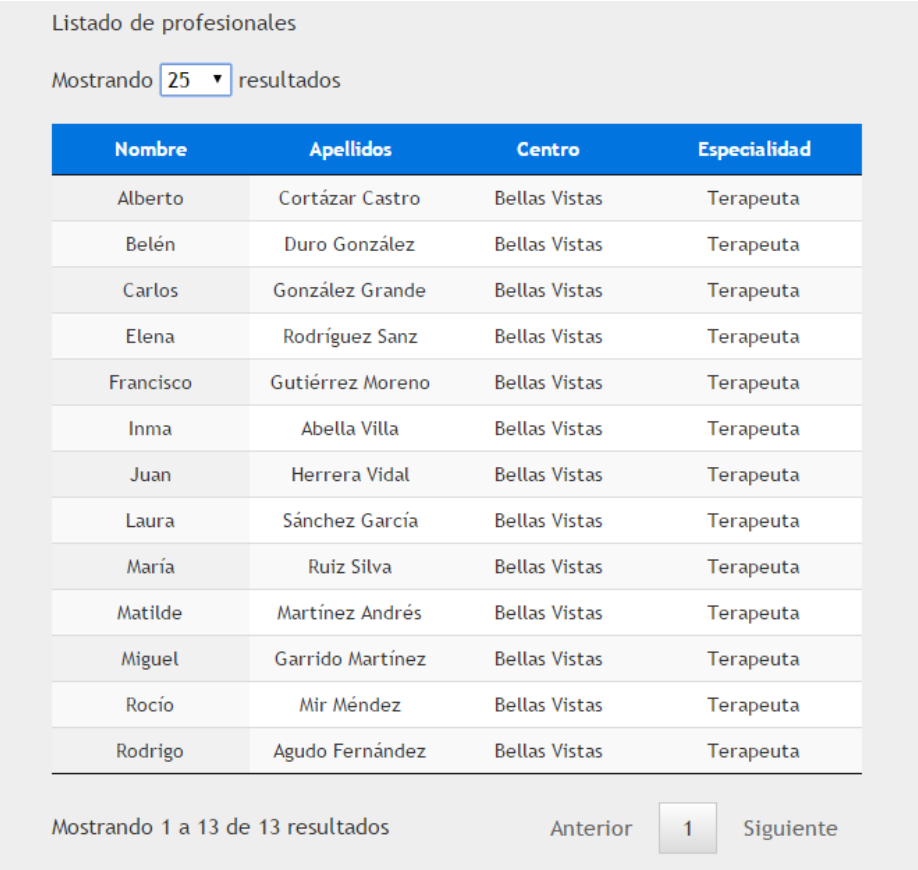


*Figura 28. Consulta de profesionales. Selección del número de resultados*

Si pulsa el botón *Consultar* desde cualquier otra opción volverá a esta tabla. Ocurre de la misma forma si se encuentra en el menú de juegos y pulsa la pestaña de *Gestión de profesionales*. Deberá utilizar la tabla de consulta, mostrada en este apartado, para modificar los datos de algún profesional o para dar de alta un nuevo profesional en el sistema.

## 8.1.Añadir un profesional

Si desea dar de alta un nuevo profesional en el sistema diríjase a la tabla consulta del menú gestión de profesionales utilizando el botón *Consultar* de este menú o la pestaña *Gestión de profesionales* y asegúrese de que ninguna fila de la tabla se encuentra seleccionada, pulsando en el espacio disponible entra la tabla y los datos. Y a continuación pulse el botón *Añadir*.

Se mostrará un menú (Figura 29) en el que debe introducir todos los datos del profesional que desea dar de alta. A continuación, pulse el botón *Enviar.*

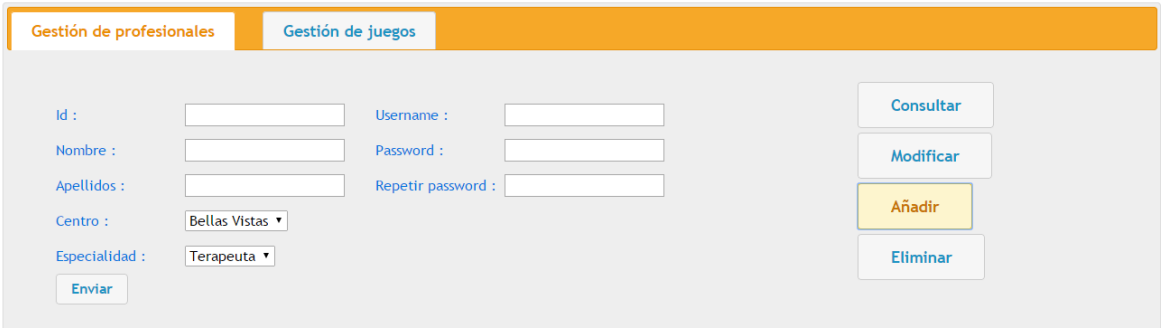


*Figura 29. Dar de alta profesional*

Una vez haya enviado la información de alta al sistema, se mostrará la tabla de consulta en la que podrá visualizar los datos del nuevo profesional.

## 8.2.Modificar un profesional

Si desea modificar los datos de un profesional registrado en el sistema, diríjase a la tabla consulta del menú gestión de profesionales (Figura 30) utilizando el botón *Consultar* o la pestaña *Gestión de profesionales*. Seleccione la fila que corresponde a los datos del profesional que desea modificar, y a continuación, pulse el botón *Modificar*. Aparecerá un formulario (Figura 31) en el que podrá visualizar los datos del profesional seleccionado.


*Figura 30. Modificar datos de profesional*


*Figura 31.Modificar datos de profesional*

Si los modifica y pulsa el botón *Actualizar datos*, se cargarán estos nuevos datos en el sistema.

### 8.3.Dar de baja a un profesional

Si desea dar de baja a alguno de los profesionales del sistema, deberá regresar a la tabla de consulta de profesionales a través del botón *Consultar* o la pestaña de *Gestión de profesionales* o el botón con el mismo nombre de la página de bienvenida.

Una vez visualice la tabla podrá seleccionar el profesional que desea dar de baja. Si no permite seleccionar pulse de nuevo el botón consultar. Una vez seleccionado si pulsa el botón eliminar se dar de baja el profesional seleccionado y se mostrará la misma tabla con los profesionales del sistema.


*Figura 32. Dar de baja profesional*

## 9. Gestión de juegos

Una vez haya pulsado el botón *Gestión de juegos* del menú principal o la pestaña con en el mismo nombre se mostrará una tabla con los juegos disponibles en el sistema. Podrá acceder a esta tabla desde cualquier otra opción de este menú, pulsando el botón *Consultar.*

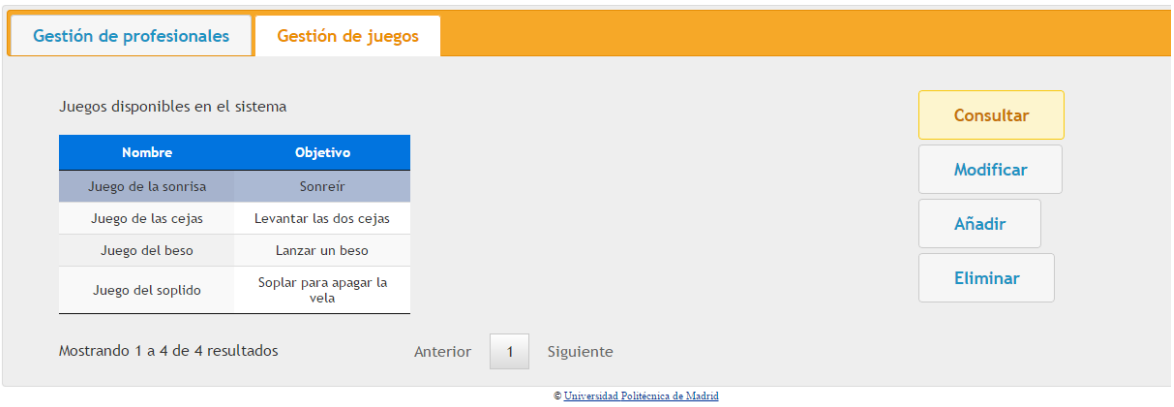


*Figura 33. Consulta de juegos*

## 9.1.Añadir un juego

Si desea dar de alta un nuevo juego en el sistema diríjase a la tabla consulta del menú gestión de juegos utilizando el botón *Consultar* de este menú o la pestaña *Gestión de juegos* y asegúrese de que ninguna fila de la tabla se encuentra seleccionada, pulsando en el espacio disponible entra la tabla y los datos. A continuación, pulse el botón *Añadir*.

Se mostrará un menú (Figura 34) en el que debe introducir todos los datos del juego que desea dar de alta. A continuación, pulse el botón *Enviar.*


*Figura 34. Dar de alta juego*

Una vez haya enviado los datos, se mostrará la tabla de consulta de este menú, en la que podrá visualizar el juego dado de alta.

## 9.2.Modificar un juego

Si desea modificar los datos de un juego registrado en el sistema, diríjase a la tabla consulta del menú Gestión de juegos utilizando el botón *Consultar* o la pestaña *Gestión de juegos*. Seleccione el juego que desea modificar, y a continuación, pulse el botón *Modificar*. Aparecerá un formulario (Figura 35) con los datos del juego seleccionado.


*Figura 35. Modificar datos de juego*

Si modifica estos datos y pulsa el botón *Actualizar datos*, se cargarán estos nuevos datos en el sistema en el juego seleccionado.

### 9.3.Dar de baja un juego

Si desea dar de baja algún juego del sistema, deberá regresar a la tabla de consulta de juegos a través del botón *Consultar*, la pestaña de *Gestión de juegos* o el botón *Gestión de juegos* de la página de bienvenida.

Una vez visualice la tabla de consulta, podrá seleccionar el juego que desea dar de baja. Si no permite seleccionar pulse de nuevo el botón consultar. Una vez seleccionado si pulsa el botón eliminar se dará de baja el juego seleccionado y se mostrará la misma tabla con los juegos que existen en el sistema.


*Figura 36. Eliminar juego del sistema*

## 10. Volver a la página de bienvenida y cerrar sesión

Podrá regresar a la página de bienvenida, pulsando el logo del sistema SONRIE tal y como se muestra en la Figura 37


*Figura 38. Volver a la página de inicio*

Si desea cerrar sesión pulse el botón *Cerrar Sesión* que se encuentra situado en la esquina superior derecha.


*Figura 39. Botón cerrar sesión*